RT–11
Mini-Reference
Manual

AA–M241A–TC

March 1983

Operating System: RT–11 Version 5.0

digital equipment corporation • maynard, massachusetts

First Printing, March 1983

Printed in U.S.A.

M17300

Contents

Introduction

This manual provides condensed information about the RT–11 monitor and utilities. The information is presented in compact form for your easy reference while working on your computer system. The mini-reference is not an explanatory guide, but rather a reminder to help you recall details of the system.

Because each RT–11 user has different reference needs, you will probably want to add your own reference pages. This binder holds regular tractor-feed line printer paper; you can print your own reference pages on your system's printer. The LA120 printer, set to print a reduced character size, is particularly effective for this purpose.

Documentation Conventions

The following documentation conventions are used in this manual:

Square brackets, if not identified as required syntax, indicate an [optional] argument

Braces { surround / contain / enclose } items from which you choose only one

Boldface type indicates default or assumed settings

Ellipses ... follow information that you can repeat

Color identifies the shortest permissible abbreviated forms of commands

lowercase letters indicate variables for which you supply an appropriate value

UPPERCASE LETTERS indicate commands which you must type as shown

The term filespec represents an RT–11 device, filename, and file type in the form dev:filnam.typ

XXX represents a special key on the terminal, for example, the space SP key

CTRL/x represents a control character, which you enter by typing the indicated key (x) while holding down the CTRL key

The term date represents a calendar date, which should be entered in the form dd:mmm:yy (for example, 18:MAR:83); if shown as optional, the default is the current system date. Dates entered as part of a Digital Command Language (DCL) command are assumed to be decimal; dates entered as arguments to utility program options require periods after the day and year to show that these numbers are decimal (for example, 18.:MAR:83.)

RT–11 Keyboard Monitor Commands

This section diagrams the valid combinations of options for each keyboard monitor command. Groups of options which can be used together but not with any others are separated from each other by a blank line. Options separated by page breaks can be used together.

ABORT (SP) jobname

The ABORT command terminates, from the system terminal, foreground and system jobs with attached private terminals.

ASSIGN (SP) physical-device-name (SP) logical-device-name

The ASSIGN command defines a logical name equivalent for a physical device.

B [(SP) address]

The B command sets a relocation base for subsequent EXAMINE or DEPOSIT commands.

BACKUP [/DEVICE /RESTORE] (SP) input-filespec (SP) output-filespec

The BACKUP command copies a single large input file or volume to several smaller output volumes.

/DEVICE	backs up a single volume to multiple volumes or restores a single volume from multiple volumes; if /DEVICE is omitted, a single large file is backed up or restored
/RESTORE	copies multiple backup volumes to a single large file or volume

BASIC

The BASIC command runs the single-user BASIC interpreter.

BOOT [/FOREIGN /WAIT] (SP) filespec

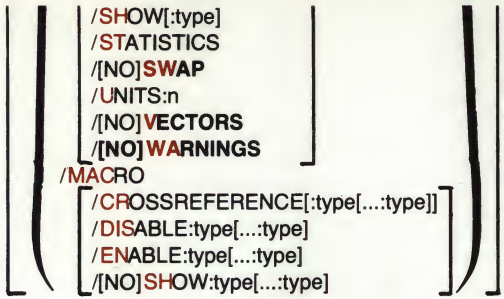
The BOOT command loads and starts a new monitor.

/FOREIGN	boots volumes which are not RT–11 V5 systems
/WAIT	waits for volume to be mounted before executing the command

CLOSE

The CLOSE command enters permanently in a volume's directory all files that are open after the background job terminates.

COMPILE
/LIST[:filespec]
/ALLOCATE:size
/[NO]OBJECT[:filespec]
/ALLOCATE:size
/DIBOL
/ALPHABETIZE
/BUFFERING
/CROSSREFERENCE
/[NO]LINENUMBERS
/LOG
/ONDEBUG
/PAGE:n
/TABLES
/[NO]WARNINGS
/FORTRAN
/CODE:type
/DIAGNOSE
/EXTEND
/HEADER
/I4
/[NO]LINENUMBERS
/ONDEBUG
/RECORD:length
SP
filespecs [/LIBRARY]



The COMPILE command runs the appropriate language processor to compile or assemble programs.

/ALLOCATE:size	reserves space for output file; a *size* of –1 reserves the largest possible space
/ALPHABETIZE	alphabetizes symbol table listing
/BUFFERING	disables double buffering
/CODE:type	controls type of code generated; *type* can be EAE, EIS, FIS, or THR
/CROSSREF[:type[...:type]]	generates cross-reference listing of program symbols; *type* can be

C — Control section names
E — Error codes
R — Register symbols
M — Macro names
P — Permanent symbols
S — User-defined symbols

no argument — equivalent to :E:M:S

/DIAGNOSE	expands crash dump information about compiler
/DIBOL	runs DIBOL compiler
/DISABLE:type[...:type]	specifies .DSABL directives; *type* can be

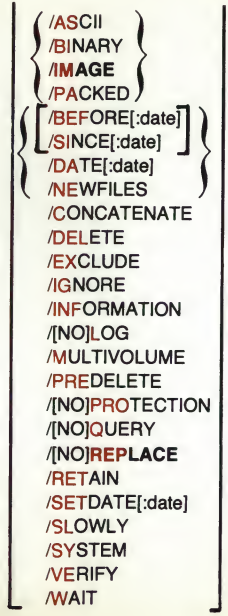
ABS — Generates absolute binary output
AMA — Assembles absolute addresses as relative addresses
CDR — Treats source columns beyond 72 as a comment
DBG — Outputs internal symbol directory (ISD) records
FPT — Truncates floating point
GBL — Assumes undefined symbols are globals
LC — Accepts lowercase characters in source programs
LSB — Defines local symbol block
MCL — Enables or disables automatic .MCALL
PNC — Enables or disables binary output
REG — Defines default register mnemonics

/ENABLE:type[...:type]	specifies .ENABL directives; *types* are listed under /DISABLE
/EXTEND	extends right mmargin from 72 to 80
/FORTRAN	runs FORTRAN compiler
/HEADER	includes list of current options in printout
/I4	allocates 2-word integers
/LIBRARY	identifies any macro library files
/[NO]LINENUMBERS	[does not] include line numbers in program
/LIST[:filespec]	outputs compiler or assembly listing to the line printer or to *filespec*

/LOG	creates log file of error messages
/MACRO	runs MACRO assembler
/[NO]OBJECT[:filespec]	[does not] generate a .OBJ file; default filename is input filename
/ONDEBUG	includes symbol names in DIBOL .OBJ file; includes debug lines in FORTRAN compilation
/PAGE:n	defines listing page size as *n* lines
/RECORD:length	overrides default record length of 132; valid range is 4 to 4095
/[NO]SHOW:type[...:type]	specifies MACRO [.NLIST] and .LIST directives; *type* can be

BEX — Extended binary code
BIN — Generated binary code
CND — Unsatisfied conditionals and .IF and .ENDC statements
COM — Comments
LOC — Location counter
LD — Listing directives with no arguments
MC — Macro calls, repeat range expansions
MD — Macro definitions, repeat range expansions
ME — Macro expansions
MEB — Macro expansions, binary code
SEQ — Source line sequence numbers
SRC — Source code
SYM — Symbol table
TOC — Table of contents
TTM — Wide or narrow listing format

/SHOW[:type]	controls FORTRAN listing format; *type* can be
	0 — diagnostics only
	1 or SRC — source program and diagnostics
	2 or MAP — storage map and diagnostics
	3 — diagnostics, source program, and storage map
	4 or COD — generated code and diagnostics
	7 or ALL — all of the above
/STATISTICS	includes compilation statistics in listing
/[NO]SWAP	[does not] allow USR to swap over program
/TABLES	includes label and symbol tables in listing
/UNITS:n	defines number of logical units; default is 6, maximum is 16
/[NO]VECTORS	[does not] access multidimensional arrays by building tables of pointers
/[NO]WARNINGS	[does not] include warning messages


/ASCII
/BINARY
/IMAGE
/PACKED
/BEFORE[:date]
/SINCE[:date]
/DATE[:date]
/NEWFILES
/CONCATENATE
/DELETE
/EXCLUDE
/IGNORE
/INFORMATION
/[NO]LOG
/MULTIVOLUME
/PREDELETE
/[NO]PROTECTION
/[NO]QUERY
/[NO]REPLACE
/RETAIN
/SETDATE[:date]
/SLOWLY
/SYSTEM
/VERIFY
/WAIT

The COPY command copies files from one location to another.

/ALLOCATE:size	reserves space for output file; a *size* of –1 reserves the largest possible space
/ASCII	copies files in ASCII mode
/BEFORE[:date]	copies files created before *date*
/BINARY	copies files in formatted binary mode
/BOOT[:dev]	copies bootstrap to volume; *dev* can be any bootable device handler name
/CONCATENATE	combines input files into single output file
/DATE[:date]	copies only files with *date*
/DELETE	deletes input file after it has been copied
/DEVICE	copies entire volume
/DOS	copies in DOS format
/END:n	specifies last block to copy
/EXCLUDE	copies all files except those you specify
/FILES	copies disk image to a file, or vice versa
/IGNORE	ignores input errors
/IMAGE	copies entire blocks
/INFORMATION	displays informational rather than fatal messages for input files not found and copies all others
/INTERCHANGE[:size]	uses interchange diskette format; *size* specifies output record size, default is 80 characters

/[NO]LOG	[does not] log names of copied files on terminal
/MULTIVOLUME	copies files from one large volume to several smaller volumes
/NEWFILES	copies only files with the current system date
/OWNER:[nnn,nnn]	specifies UIC for DOS file transfers; square brackets are part of syntax
/PACKED	copies files from DOS, TOPS, or INTERCHANGE
/POSITION:n	positions magtape (refer to PIP /M:n in Utility Program Options section)
/PREDELETE	deletes files of same name on output volume before copying the input files
/[NO]PROTECTION	[does not] protect copied files
/[NO]QUERY	[does not] ask for confirmation before copying each file
/[NO]REPLACE	[does not] replace existing file of same name on output volume
/RETAIN	preserves output volume's bad block replacement table
/SETDATE[:date]	assigns *date* to output files
/SINCE[:date]	copies files created on or after *date*
/SLOWLY	transfers files one block at a time
/START:n	specifies first block to copy
/SYSTEM	includes .SYS files in copy
/TOPS	specifies DECsystem–10 DECtape input
/VERIFY	compares copied output to input
/WAIT	waits for volume to be mounted before executing the command

CREATE (SP) filespec /EXTENSION:n
/START:n
/ALLOCATE:size

The CREATE command creates or extends a file entry in a volume's directory.

/ALLOCATE:size	specifies the size of the file to create; a *size* of –1 reserves the largest possible space
/EXTENSION:n	adds *n* blocks to an existing file; at least *n* free blocks must follow the file
/START:n	specifies *n* as the starting block of the file being created

D (SP) address=value[,...value]

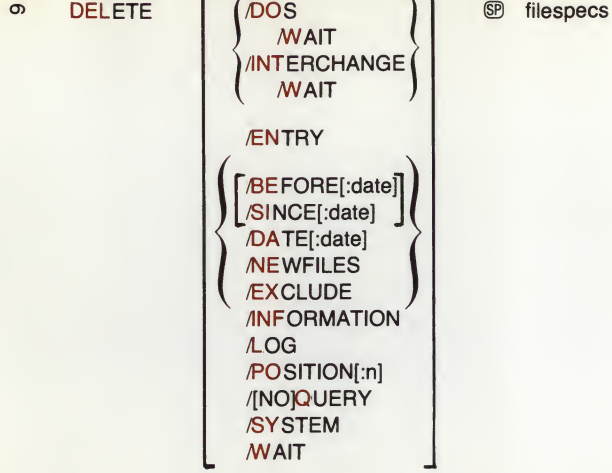
The D command deposits values in memory beginning at the specified address.

DATE [Ⓢ dd-mmm-yy]

The DATE command sets or displays the system date.

DEASSIGN Ⓢ logical-device-name

The DEASSIGN command removes logical name assignments established with ASSIGN. DEASSIGN with no argument removes all assignments that are currently in effect.


DELETE
/DOS
/WAIT
/INTERCHANGE
/WAIT
/ENTRY
/BEFORE[:date]
/SINCE[:date]
/DATE[:date]
/NEWFILES
/EXCLUDE
/INFORMATION
/LOG
/POSITION[:n]
/[NO]QUERY
/SYSTEM
/WAIT
SP
filespecs

The DELETE command deletes files from a volume's directory or from the system queue.

/BEFORE[:date]	deletes files created before *date*
/DATE[:date]	deletes files created on *date*
/DOS	deletes files from DOS or RSTS/E disks or DECtapes
/ENTRY	deletes a job from the system queue
/EXCLUDE	deletes all files except those you specify
/INFORMATION	displays informational rather than fatal messages for files not found and deletes all others
/INTERCHANGE	deletes files from an interchange format diskette
/LOG	lists the names of all deleted files on the console terminal
/NEWFILES	deletes only files with the current system date
/POSITION[:n]	controls positioning of tapes for deletions

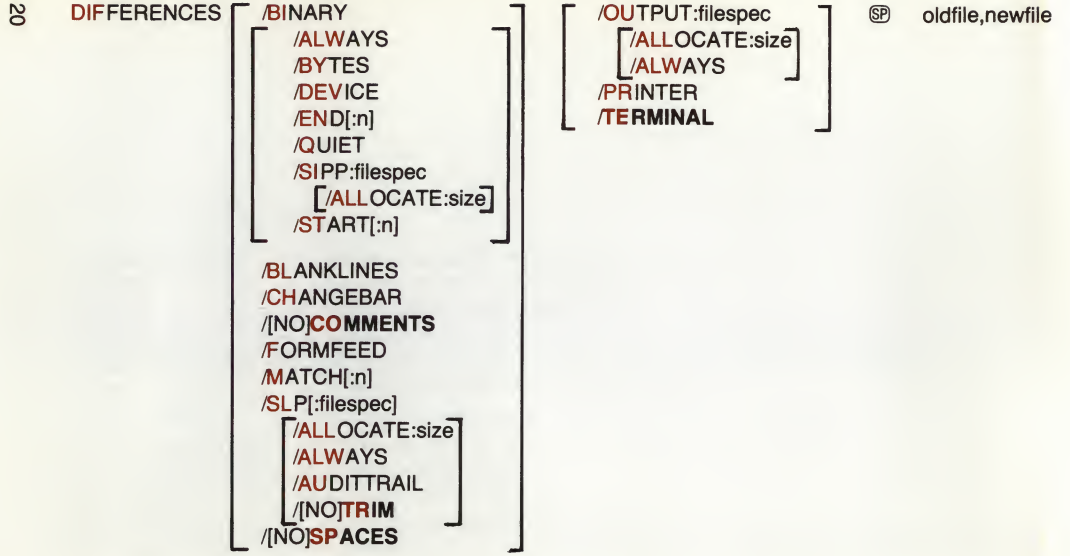
n = 0 rewinds before searching for file
n = +n searches forward from current position for specified file; if file is not found before the *n*th file, the *n*th file is deleted
n = −n rewinds the tape, then proceeds as for +*n*, above

/[NO]QUERY	[does not] ask for confirmation before deleting each file
/SINCE[:date]	deletes files created on or after *date*
/SYSTEM	permits deletion of .SYS files
/WAIT	waits for volume to be mounted before executing the command

```
DIBOL  [ /ALPHABETIZE                ]   (SP) filespecs
       [ /BUFFERING                  ]
       [ /CROSSREFERENCE             ]
       [ /[NO]LINENUMBERS            ]
       [ /LIST[:filespec]            ]
       [    /ALLOCATE:size           ]
       [ /LOG                        ]
       [ /[NO]OBJECT[:filespec]      ]
       [    /ALLOCATE:size           ]
       [ /ONDEBUG                    ]
       [ /PAGE:n                     ]
       [ /TABLES                     ]
       [ /[NO]WARNINGS               ]
```

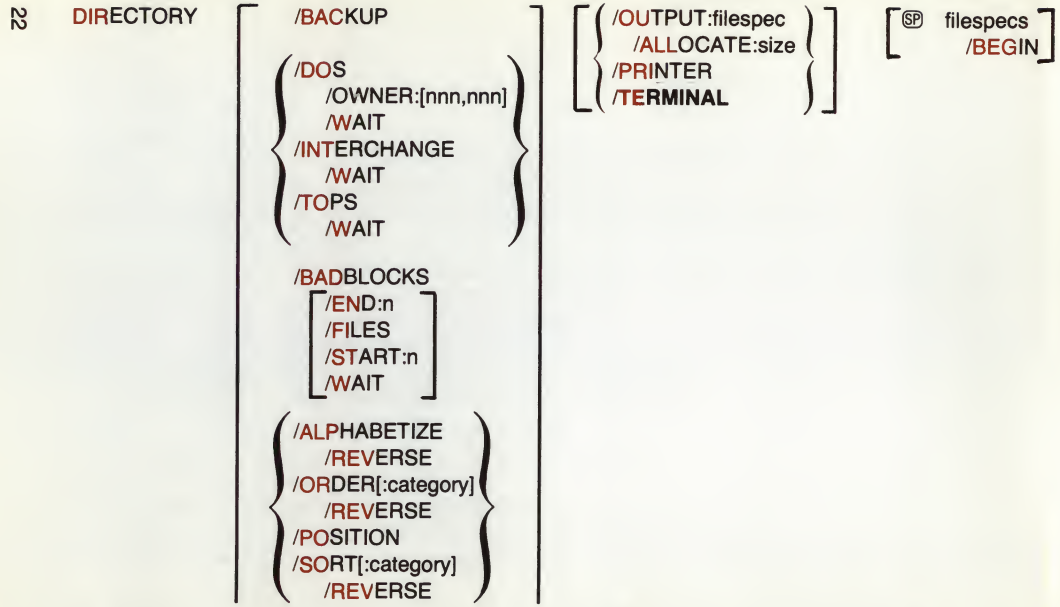
The DIBOL command runs the DIBOL compiler.

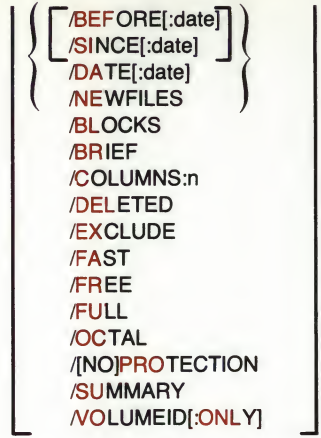
/ALLOCATE:size	reserves space for output file; a *size* of –1 reserves the largest possible space
/ALPHABETIZE	alphabetizes symbol table listing
/BUFFERING	disables double buffering
/CROSSREFERENCE	outputs CREF listing
/[NO]LINENUMBERS	[does not] include line numbers in executable program
/LIST[:filespec]	outputs program listing to the line printer or to *filespec*
/LOG	outputs log file of compiler error messages
/[NO]OBJECT[:filespec]	[does not] produce .OBJ file; default filename is input filename
/ONDEBUG	includes symbol table in .OBJ file for use with DIBOL DDT
/PAGE:n	defines listing page size as *n* lines
/TABLES	includes label and symbol tables in listing
/[NO]WARNINGS	[does not] include warning messages


DIFFERENCES
/BINARY
/ALWAYS
/BYTES
/DEVICE
/END[:n]
/QUIET
/SIPP:filespec
/ALLOCATE:size
/START[:n]
/BLANKLINES
/CHANGEBAR
/[NO]COMMENTS
/FORMFEED
/MATCH[:n]
/SLP[:filespec]
/ALLOCATE:size
/ALWAYS
/AUDITTRAIL
/[NO]TRIM
/[NO]SPACES
/OUTPUT:filespec
/ALLOCATE:size
/ALWAYS
/PRINTER
/TERMINAL
SP
oldfile,newfile

The DIFFERENCES command compares two files and lists the differences between them.

/ALLOCATE:size	reserves space for output file; a *size* of –1 reserves the largest possible space
/ALWAYS	creates output file whether or not differences exist
/AUDITTRAIL	marks any changes made by SLP
/BINARY	compares binary files
/BLANKLINES	includes blank lines when comparing
/BYTES	lists differences byte-by-byte
/CHANGEBAR	marks additions and deletions in new file
/[NO]COMMENTS	[does not] include comments in comparison
/DEVICE	compares two entire volumes
/END[:n]	specifies last block to compare
/FORMFEED	includes formfeeds in output listing
/MATCH[:n]	specifies the number of lines that constitute a match; default is 3, valid range is 1–200
/OUTPUT:filespec	specifies output device and filename for differences listing
/PRINTER	prints differences on the line printer
/QUIET	suppresses listing of differences on the console terminal
/SIPP:filespec	creates a command file for SIPP
/SLP:filespec	creates a command file for SLP
/[NO]SPACES	[does not] include spaces and tabs in comparison
/START[:n]	specifies first block to compare
/TERMINAL	lists differences on the console terminal
/[NO]TRIM	[does not] ignore tabs and spaces at the ends of lines


DIRECTORY
/BACKUP
/DOS
/OWNER:[nnn,nnn]
/WAIT
/INTERCHANGE
/WAIT
/TOPS
/WAIT
/BADBLOCKS
/END:n
/FILES
/START:n
/WAIT
/ALPHABETIZE
/REVERSE
/ORDER[:category]
/REVERSE
/POSITION
/SORT[:category]
/REVERSE
/OUTPUT:filespec
/ALLOCATE:size
/PRINTER
/TERMINAL
SP
filespecs
/BEGIN



The DIRECTORY command lists volume directories.

/ALLOCATE:size	reserves space for output file; a *size* of –1 reserves the largest possible space
/ALPHABETIZE	lists directory entries in alphabetical order
/BACKUP	lists directory of a volume created by BUP
/BADBLOCKS	scans volume for bad blocks and lists their block numbers

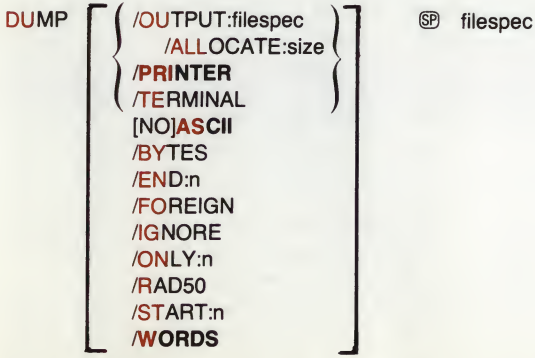
/BEFORE[:date]	lists files created before *date*
/BEGIN	lists directory beginning at file you specify
/BLOCKS	includes starting block numbers of files and free areas in directory listing
/BRIEF	lists only file name and type; equivalent to /FAST
/COLUMNS:n	lists directory in *n*-column format
/DATE[:date]	lists only files created on *date*
/DELETED	lists files that have been deleted
/DOS	lists directory of DOS-format volume
/END:n	specifies final block for a bad block scan
/EXCLUDE	lists all files except those you specify
/FAST	lists only file name and type; equivalent to /BRIEF
/FILES	prints names of files containing bad blocks
/FREE	lists sizes of all free areas
/FULL	lists file names, free areas, sizes, and creation dates
/INTERCHANGE	lists directory of interchange diskette
/NEWFILES	lists only those files with the current system date
/OCTAL	lists in octal sizes and, if /BLOCK is used, starting blocks
/ORDER[:category]	sorts the directory listing by category; category can be

DATE — creation date SIZE — file size
NAME — file name TYPE — file type
POSITION — file position on the volume

/OUTPUT:filespec	specifies output device and filename for directory listing
/OWNER:[nnn,nnn]	specifies UIC for DOS-format volume; square brackets are part of syntax
/POSITION	lists files in the order that they occur on the volume; includes file sequence numbers of files on magtape or starting block numbers of files on disk
/PRINTER	prints directory listing on the line printer
/[NO]PROTECTION	lists all [un]protected files on volume
/REVERSE	sorts directory in reverse order
/SINCE[:date]	lists all files created on or after *date*
/SORT[:category]	equivalent to /ORDER
/START:n	specifffies starting block for bad block scan
/SUMMARY	lists number of files, blocks in use, and free blocks on volume
/TERMINAL	lists directory on the console terminal
/TOPS	lists directory of DECsystem–10 DECtape
/VOLUMEID[:ONLY]	includes volume ID and owner name at beginning of directory listing; with *ONLY*, lists just volume ID and owner name
/WAIT	waits for volume to be mounted before executing the command

DISMOUNT (SP) logical-disk-unit

The DISMOUNT command frees a logical disk unit number from its associated file.

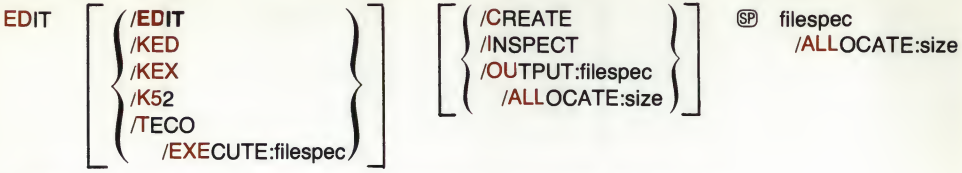


The DUMP command lists the contents of files in various formats.

/ALLOCATE:size	reserves space for the output file; a *size* of –1 reserves the largest possible space
/[NO]ASCII	[does not] list ASCII equivalent of each octal word or byte
/BYTES	lists information in octal bytes
/END:n	specifies last block to dump
/FOREIGN	dumps a magtape that is not RT–11 format
/IGNORE	ignores I/O errors
/ONLY:n	dumps only specified block
/OUTPUT:filespec	sends output to *filespec*
/PRINTER	sends output to the line printer
/RAD50	lists RAD50 equivalent of each octal word
/START:n	specifies first block to dump
/TERMINAL	sends output to console terminal
/WORDS	lists information in octal words

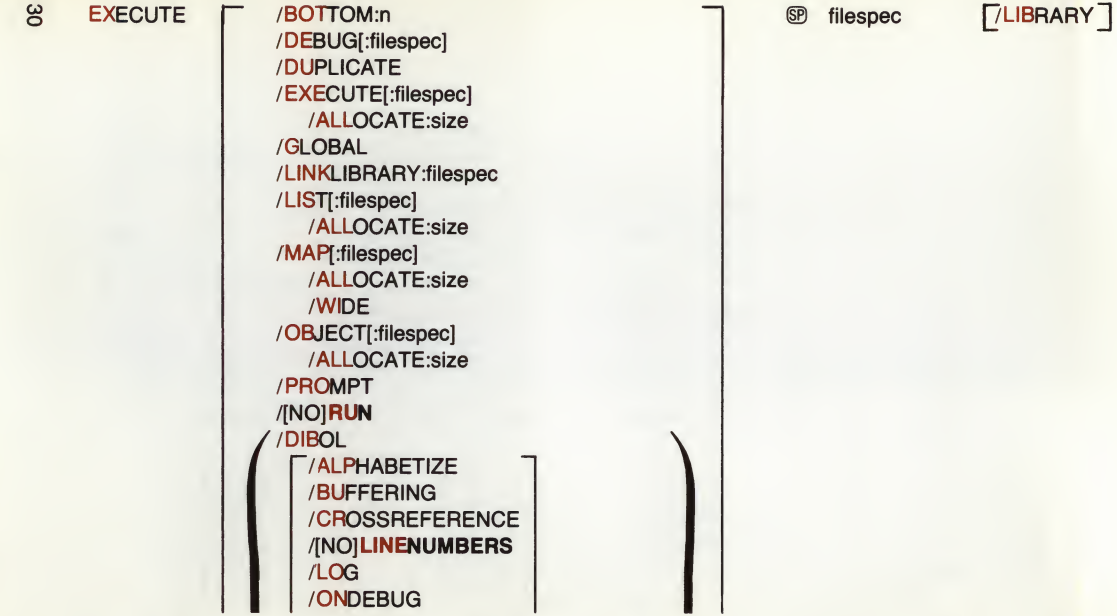
E (SP) address[–address]

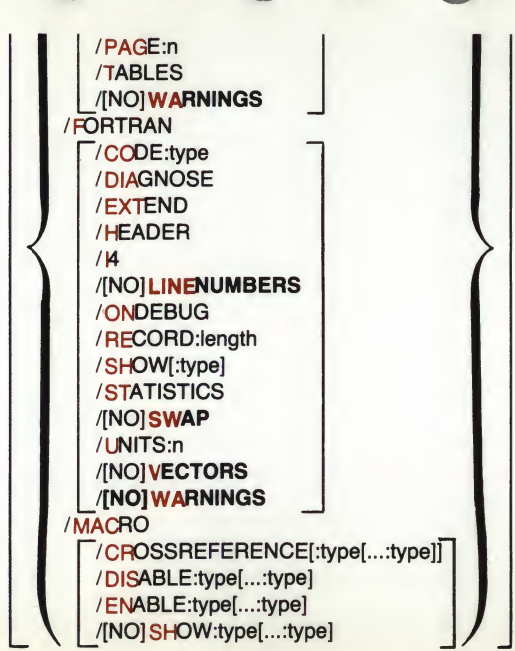
The E command examines memory and displays the contents on the console terminal.



The EDIT command runs a text editor for creating or modifying ASCII files.

/ALLOCATE:size	reserves space for the output file; a *size* of –1 reserves the largest possible space
/CREATE	creates a new file
/EDIT	runs the RT–11 editor for hardcopy terminals
/EXECUTE:filespec	executes TECO commands from a file
/INSPECT	opens file as read-only
/KED	runs the KED editor for VT100 terminals
/KEX	runs the KED editor for VT100 terminals under the XM monitor
/K52	runs the K52 editor for VT52 terminals
/OUTPUT:filespec	directs edited output to a file
/TECO	runs the TECO editor


EXECUTE
/BOTTOM:n
/DEBUG[:filespec]
/DUPLICATE
/EXECUTE[:filespec]
/ALLOCATE:size
/GLOBAL
/LINKLIBRARY:filespec
/LIST[:filespec]
/ALLOCATE:size
/MAP[:filespec]
/ALLOCATE:size
/WIDE
/OBJECT[:filespec]
/ALLOCATE:size
/PROMPT
/[NO]RUN
/DIBOL
/ALPHABETIZE
/BUFFERING
/CROSSREFERENCE
/[NO]LINENUMBERS
/LOG
/ONDEBUG
SP
filespec
/LIBRARY


/PAGE:n
/TABLES
/[NO]WARNINGS
/FORTRAN
/CODE:type
/DIAGNOSE
/EXTEND
/HEADER
/I4
/[NO]LINENUMBERS
/ONDEBUG
/RECORD:length
/SHOW[:type]
/STATISTICS
/[NO]SWAP
/UNITS:n
/[NO]VECTORS
/[NO]WARNINGS
/MACRO
/CROSSREFERENCE[:type[...:type]]
/DISABLE:type[...:type]
/ENABLE:type[...:type]
/[NO]SHOW:type[...:type]

The EXECUTE command compiles or assembles source files, then links and runs the resulting modules.

/ALLOCATE:size — reserves space for output file; a *size* of –1 reserves the largest possible space

/ALPHABETIZE — alphabetizes symbol table listing

/BOTTOM:n — specifies lowest address to be used by the relocatable code in load module

/BUFFERING — disables double buffering

/CODE:type — controls type of code generated; *type* can be EAE, EIS, FIS, or THR

/CROSSREFERENCE[:type[...:type]] — generates cross-reference listing of program symbols; *type* can be

C — Control section names
M — Macro names
E — Error codes
P — Permanent symbols
R — Register symbols
S — User-defined symbols
no argument — equivalent to :E:M:S

/DEBUG[:filespec] — links ODT [or another debugger] with program

/DIAGNOSE — expands crash dump information about compiler

/DIBOL — runs DIBOL compiler

/DISABLE:type[...:type] — specifies .DSABL directives; *type* can be

ABS — Generates absolute binary output
AMA — Assembles absolute addresses as relative addresses
CDR — Treats source columns beyond 72 as a comment

DBG — Outputs internal symbol directory (ISD) records
FPT — Truncates floating point
GBL — Assumes undefined symbols are globals
LC — Accepts lowercase characters in source programs
LSB — Defines local symbol block
MCL — Enables or disables automatic .MCALL
PNC — Enables or disables binary output
REG — Defines default register mnemonics

/DUPLICATE	allows multiple copies of library modules when linking
/ENABLE:type[...:type]	specifies .ENABL directives; *types* are listed under /DISABLE
/EXECUTE[:filespec]	specifies *filespec* for executable file; default filename is input filename
/EXTEND	extends right margin of input lines from column 72 to column 80
/FORTRAN	runs FORTRAN compiler
/GLOBAL	includes cross reference of global symbols as part of load map listing
/HEADER	includes list of current options in printout
/I4	allocates 2-word integers
/LIBRARY	identifies any macro library files
/[NO]LINENUMBERS	[does not] include line numbers in executable program
/LINKLIBRARY:filespec	includes a file as a library at link time
/LIST[:filespec]	outputs compilation or assembly listing to the line printer or to *filespec*
/LOG	outputs a log file of compiler error messages
/MACRO	runs MACRO assembler
/MAP[:filespec]	outputs load map to the line printer or to *filespec*

/OBJECT[:filespec]	specifies file name for .OBJ file
/ONDEBUG	includes symbol names in output file for use with DIBOL DDT; includes debug lines in FORTRAN compilation
/PAGE:n	defines listing page size as *n* lines
/PROMPT	accepts input lines until a terminating double slash (//) is found
/RECORD:length	overrides default record length of 132 characters; valid range is 4 to 4095
/[NO]RUN	[does not] run the program after linking
/[NO]SHOW:type[...:type]	specifies MACRO [.NLIST] and .LIST directives; *type* can be

BEX — Extended binary code
BIN — Generated binary code
CND — Unsatisfied conditionals and .IF and .ENDC statements
COM — Comments
LOC — Location counter
LD — Listing directives with no arguments
MC — Macro calls, repeat range expansions
MD — Macro definitions, repeat range expansions
ME — Macro expansions
MEB — Macro expansions, binary code
SEQ — Source line sequence numbers
SRC — Source code
SYM — Symbol table
TOC — Table of contents
TTM — Wide or narrow listing format

/SHOW[:type]	controls FORTRAN listing format; *type* can be 0 — diagnostics only 1 or SRC — source program and diagnostics 2 or MAP — storage map and diagnostics **3** — diagnostics, source program, and storage map 4 or COD — generated code and diagnostics 7 or ALL — all of the above
/STATISTICS	includes compilation statistics in listing
/[NO]SWAP	[does not] permit USR to swap over FORTRAN program
/TABLES	includes label and symbol tables in listing
/UNITS:n	defines number of FORTRAN logical units; initial value of *n* is 6, maximum is 16
/[NO]VECTORS	[does not] access FORTRAN multidimensional arrays by building tables of pointers
/[NO]WARNINGS	[does not] include warning messages
/WIDE	outputs 132-column load map listing

FORMAT [/[NO]QUERY /SINGLEDENSITY /VERIFY[:ONLY] /PATTERN:value /WAIT] (SP) device

The FORMAT command formats the following disks and diskettes:

RK05
RK06–RK07
RX01–RX02

In addition, with the /VERIFY:ONLY option, the command can verify any RT–11 disk, diskette, or DECtape II, except an MSCP class disk.

/PATTERN[:value] specifies which of the following bit patterns to use for verification:

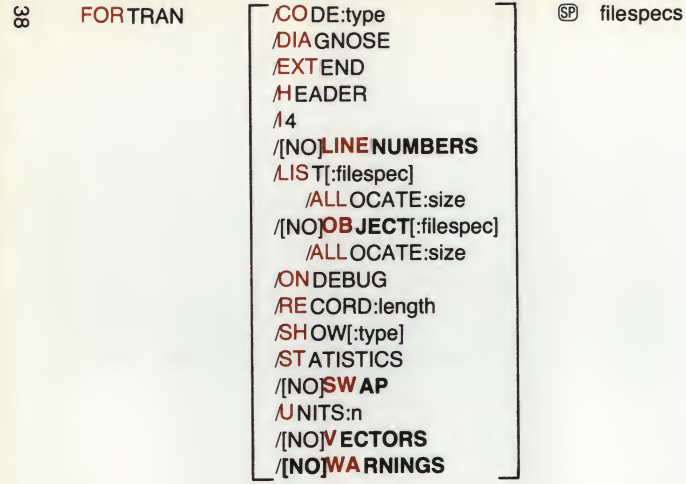
value	pattern	value	pattern
1	000000	100	021042
2	111111	**200**	104210
4	163126	400	155555
10	125252	1000	145454
20	052525	2000	146314
40	007417		

/[NO]QUERY [does not] ask for confirmation before formatting

/SINGLEDENSITY formats diskettes in single density format

/VERIFY[:ONLY] verifies after formatting; with *ONLY*, verifies without formatting

/WAIT waits for volume to be mounted before executing the command


FORTRAN
/CODE:type
/DIAGNOSE
/EXTEND
/HEADER
/I4
/[NO]LINENUMBERS
/LIST[:filespec]
/ALLOCATE:size
/[NO]OBJECT[:filespec]
/ALLOCATE:size
/ONDEBUG
/RECORD:length
/SHOW[:type]
/STATISTICS
/[NO]SWAP
/UNITS:n
/[NO]VECTORS
/[NO]WARNINGS
SP
filespecs

The FORTRAN command runs the FORTRAN compiler.

/ALLOCATE:size	reserves space for output file; a *size* of –1 reserves the largest possible space
/CODE:type	controls type of code generated; *type* can be EAE, EIS, FIS, or THR
/DIAGNOSE	expands crash dump information about compiler
/EXTEND	extends right margin of input lines from 72 to 80 columns
/HEADER	includes list of current options in printout
/I4	allocates two-word integers
/[NO]LINENUMBERS	[does not] include line numbers in executable program
/LIST[:filespec]	outputs compiler listing to the line printer or to *filespec*
/[NO]OBJECT[:filespec]	[does not] generate an OBJ file
/ONDEBUG	includes debug lines (those with D in column 1) in the compilation
/RECORD:length	overrides default record length of 132 characters; valid range is 4 to 4095
/SHOW[:type]	controls FORTRAN listing format; *type* can be

0	— diagnostics only
1 or SRC	— source program and diagnostics
2 or MAP	— storage map and diagnostics
3	— diagnostics, source program, and storage map
4 or COD	— generated code and diagnostics
7 or ALL	— all of the above

/STATISTICS	includes compilation statistics in listing
/[NO]SWAP	[does not] allow USR to swap over FORTRAN program
/UNITS:n	defines number of FORTRAN logical units; initial value of *n* is 6, maximum is 16
/[NO]VECTORS	[does not] access FORTRAN multidimensional arrays by building tables of pointers
/[NO]WARNINGS	[does not] include warning messages

FRUN Ⓢ filespec [/BUFFER:n /NAME:name /PAUSE /TERMINAL:n]

The FRUN command runs foreground jobs.

/BUFFER:n	reserves more space in memory than actual program size
/NAME:name	assigns a logical name to the foreground job
/PAUSE	prints load address of foreground job and waits for RESUME
/TERMINAL:n	assigns terminal n to the foreground job

GET Ⓢ filespec

The GET command loads a memory image file into memory.

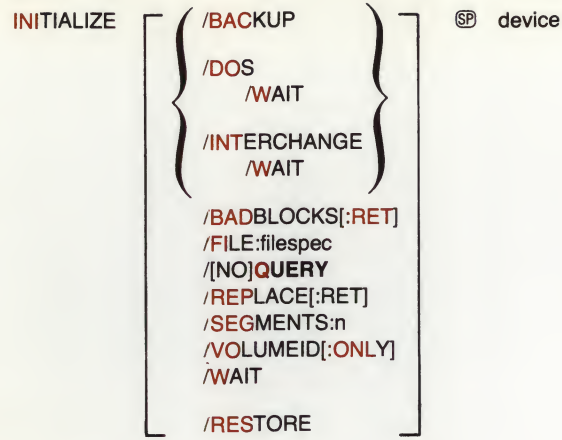
GT (SP) { OFF | ON | [/L:n] [/T:n] }

The GT command enables or disables the VT–11 or VS60 display hardware.

/L:n	sets the number of lines of text displayed at one time
/T:n	sets the top position of the scroll display

The HELP command lists information about various topics and options.

/PRINTER	lists information on the line printer
/TERMINAL	lists information on the console terminal

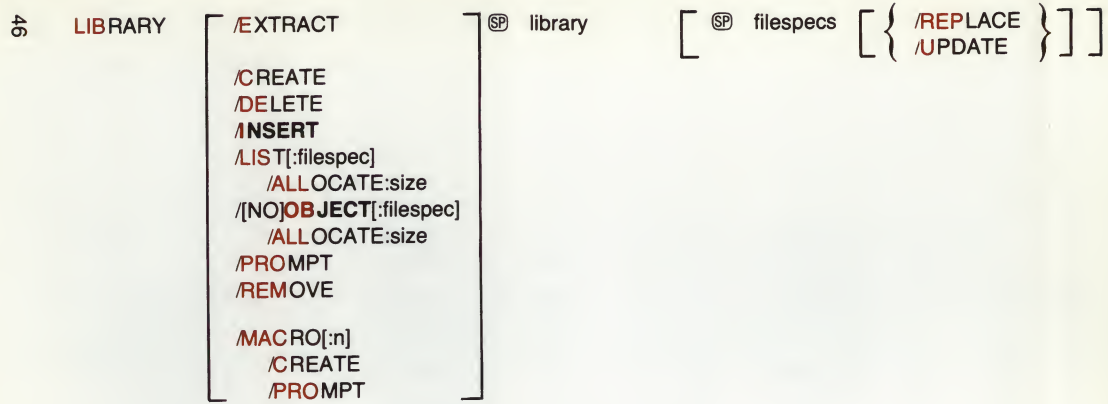

INITIALIZE
/BACKUP
/DOS
/WAIT
/INTERCHANGE
/WAIT
/BADBLOCKS[:RET]
/FILE:filespec
/[NO]QUERY
/REPLACE[:RET]
/SEGMENTS:n
/VOLUMEID[:ONLY]
/WAIT
/RESTORE
SP
device

The INITIALIZE command writes an empty directory on a volume.

/BACKUP	initializes a volume for use with BUP
/BADBLOCKS[:RET]	scans the volume for bad blocks and creates FILE.BAD directory entries; with *RET*, retains existing FILE.BAD files but does not scan for bad blocks
/DOS	initializes a DOS-format DECtape
/FILE:filespec	initializes a bootable magtape
/INTERCHANGE	initializes an interchange diskette
/[NO]QUERY	[does not] ask for confirmation before initialization
/REPLACE[:RET]	creates [retains the old] bad block replacement table
/RESTORE	restores old directory on initialized volume
/SEGMENTS:n	specifies the number of directory segments to allocate
/VOLUMEID[:ONLY]	asks for owner name and volume ID during initialization; with *ONLY*, writes a new owner name and volume ID without changing the directory
/WAIT	waits for volume to be mounted before executing the command

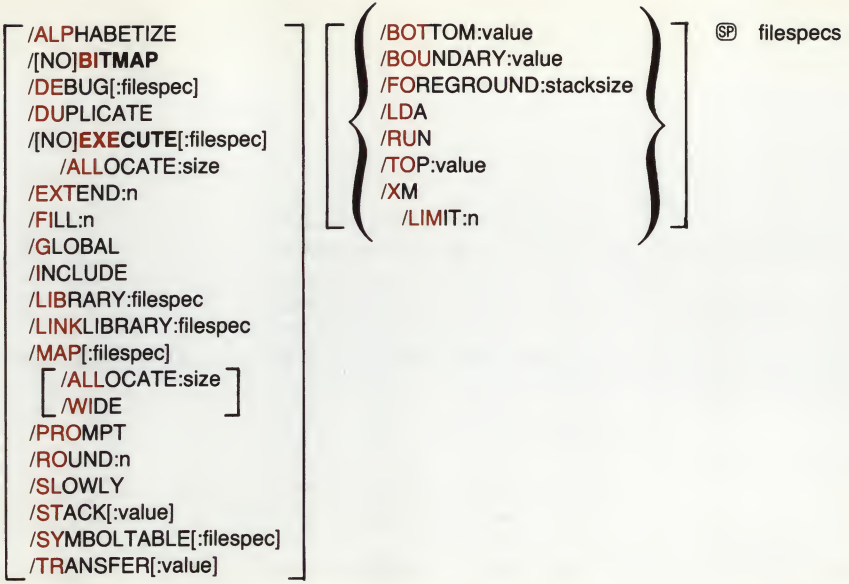
INSTALL (SP) device[,...device]

The INSTALL command enters device names in the monitor system tables.


LIBRARY
/EXTRACT
/CREATE
/DELETE
/INSERT
/LIST[:filespec]
/ALLOCATE:size
/[NO]OBJECT[:filespec]
/ALLOCATE:size
/PROMPT
/REMOVE
/MACRO[:n]
/CREATE
/PROMPT
SP
library
SP
filespecs
/REPLACE
/UPDATE

The LIBRARY command creates and updates .OBJ and macro libraries.

/ALLOCATE:size	reserves space for the output file; a *size* of –1 reserves the largest possible space
/CREATE	creates a new .OBJ library
/DELETE	deletes an .OBJ module from a library
/EXTRACT	retrieves an .OBJ module from a library and puts it in a separate file
/INSERT	adds an .OBJ module to an existing library
/LIST[:filespec]	outputs a directory listing of an .OBJ library to the line printer or to *filespec*
/MACRO[:n]	creates a macro library; *n* is the size in blocks of the macro name directory
/[NO]OBJECT[:filespec]	[does not] create a new .OBJ library from an old library; output filename defaults to input filename
/PROMPT	allows more than one line of input files; terminate with //
/REMOVE	deletes global symbols from library directory
/REPLACE	replaces existing modules with modules of the same name
/UPDATE	combines /INSERT and /REPLACE


LINK
/ALPHABETIZE
/[NO]BITMAP
/DEBUG[:filespec]
/DUPLICATE
/[NO]EXECUTE[:filespec]
/ALLOCATE:size
/EXTEND:n
/FILL:n
/GLOBAL
/INCLUDE
/LIBRARY:filespec
/LINKLIBRARY:filespec
/MAP[:filespec]
/ALLOCATE:size
/WIDE
/PROMPT
/ROUND:n
/SLOWLY
/STACK[:value]
/SYMBOLTABLE[:filespec]
/TRANSFER[:value]
/BOTTOM:value
/BOUNDARY:value
/FOREGROUND:stacksize
/LDA
/RUN
/TOP:value
/XM
/LIMIT:n
SP
filespecs

The LINK command combines .OBJ modules into a program that can be run.

/ALLOCATE:size	reserves space for the output file; a *size* of –1 reserves the largest possible space
/ALPHABETIZE	lists globals in load map in alphabetical order
/[NO]BITMAP	[does not] create a memory usage bitmap
/BOTTOM:value	specifies lowest address to be used by relocatable code in load module; invalid with /TOP and /FOREGROUND
/BOUNDARY:value	links program section starting at a specified address boundary; invalid with /TOP
/DEBUG[:filespec]	links ODT [or another debugger] with program
/DUPLICATE	allows multiple copies of a library or module
/EXECUTE[:filespec]	specifies name for LINK output file; output filename defaults to input filename
/NOEXECUTE	suppresses creation of an output file
/EXTEND:n	extends a program section
/FILL:n	initializes unused locations to the value n
/FOREGROUND[:stacksize]	links file in .REL format; with *stacksize*, specifies stack size; invalid with /BOTTOM, /TOP, /LIMIT, and /LDA

/GLOBAL	produces cross-reference of all global symbols in the load map
/INCLUDE	asks for globals to be included from libraries
/LDA	links file in LDA format; invalid with /FOREGROUND and /XM
/LIBRARY:filespec	same as /LINKLIBRARY
/LIMIT:n	specifies high limit for XM virtual .SETTOP; valid only with /XM; *n* is K-word value less than or equal to 32(decimal)
/LINKLIBRARY:filespec	identifies library files
/MAP[:filespec]	outputs load map to the line printer or to *filespec*
/PROMPT	allows multiple input lines; terminate with //
/ROUND:n	rounds section size up to a multiple of *n*; *n* must be a power of 2
/RUN	after linking, runs the resulting .SAV file; invalid with /FOREGROUND and /LDA
/SLOWLY	allows largest possible LINK symbol table
/STACK[:value]	sets stack address in location 42 to *value*
/SYMBOLTABLE[:filespec]	creates a file of global symbol definitions
/TOP:value	specifies highest address to be used; invalid with /BOTTOM and /FOREGROUND
/TRANSFER[:value]	specifies starting address for execution
/WIDE	produces wide link map
/XM	enables virtual .SETTOP; invalid with /LDA

LOAD (SP) device[=jobname][,...device[=jobname]]

The LOAD command fetches device handlers and assigns them to jobs.

MACRO [/CROSSREFERENCE[:type[...:type]] | /DISABLE:type[...:type] | /ENABLE:type[...:type] | /LIST[:filespec] /ALLOCATE:size | /[NO]OBJECT[:filespec] /ALLOCATE:size | /[NO]SHOW:type[...:type]] (SP) filespecs [/LIBRARY]

The MACRO command runs the MACRO assembler.

/ALLOCATE:size — reserves space for output file; a *size* of –1 reserves the largest possible space

/CROSSREFERENCE[:type[...:type]] produces CREF listing; *type* can be

C — Control section names
M — Macro names
E — Error codes
P — Permanent symbols
R — Register symbols
S — User-defined symbols
no argument — equivalent to :E:M:S

/DISABLE:type[...:type] — specifies .DSABL directives; *type* can be

ABS — Produces absolute binary output
AMA — Assembles absolute addresses as relative addresses
CDR — Treats source columns beyond 72 as a comment
DBG — Outputs internal symbol directory (ISD) records
FPT — Truncates floating point
GBL — Assumes undefined symbols are globals
LC — Accepts lower case characters in source programs
LSB — Defines local symbol block
MCL — Enables or disables automatic .MCALL
PNC — Enables or disables binary output
REG — Defines default register mnemonics

/ENABLE:type[...:type]	specifies .ENABL directives; *types* are listed under /DISABLE
/LIBRARY	identifies a macro library file
/LIST[:filespec]	outputs program listing to the line printer or to *filespec*
/[NO]OBJECT[:filespec]	[does not] generate an .OBJ file; output filename defaults to input filename
/[NO]SHOW:type[...:type]	specifies MACRO [.NLIST] and .LIST directives; *type* can be

BEX — Extended binary code
BIN — Generated binary code
CND — Unsatisfied conditionals and .IF and .ENDC statements
COM — Comments
LOC — Location counter
LD — Listing directives with no arguments
MC — Macro calls, repeat range expansions
MD — Macro definitions, repeat range expansions
ME — Macro expansions
MEB — Macro expansions, binary code
SEQ — Source line sequence numbers
SRC — Source code
SYM — Symbol table
TOC — Table of contents
TTM — Wide or narrow listing format

MAKE (SP) filespec

The MAKE command is available only under the FB amd XM monitors. It is equivalent to:

```
.R TECO
*EWfilespec$$
```

MOUNT /[NO]**WRITE** (SP) logical-disk-unit (SP) filespec [(SP) logical-device-name]

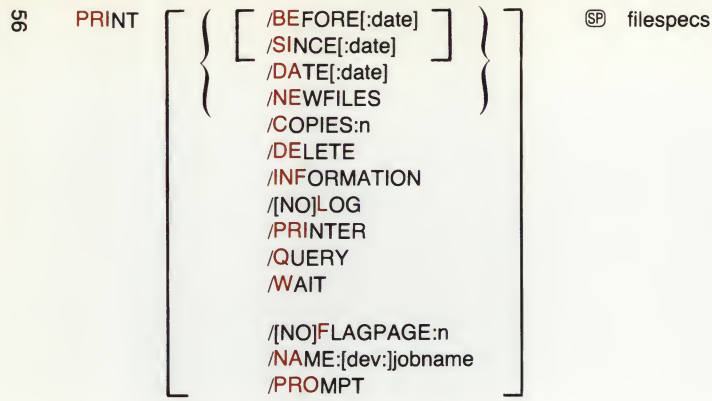
The MOUNT command associates a logical disk unit with a logical disk file and optionally assigns it a logical name.

/[NO]WRITE [does not] permit writing to the logical disk unit

MUNG (SP) filespec[,data]

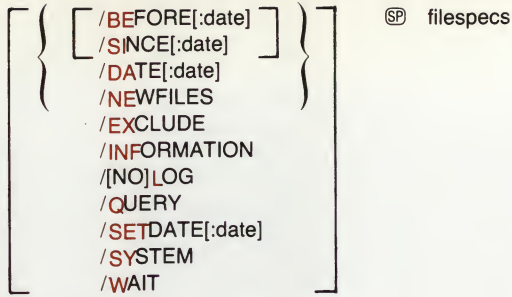
The MUNG command is available only under the FB and XM monitors. It is equivalent to:

```
.R TECO
[Idata$]EIfilespec$$
```


PRINT
/BEFORE[:date]
/SINCE[:date]
/DATE[:date]
/NEWFILES
/COPIES:n
/DELETE
/INFORMATION
/[NO]LOG
/PRINTER
/QUERY
/WAIT
/[NO]FLAGPAGE:n
/NAME:[dev:]jobname
/PROMPT
SP
filespecs

The PRINT command prints the contents of one or more files on the line printer.

/BEFORE[:date]	prints files created before *date*
/COPIES:n	prints *n* copies of a file; *n* can be from 2 to 32
/DATE[:date]	prints files created on *date*
/DELETE	deletes a file after printing
/[NO]FLAGPAGE:n	valid only when QUEUE is running; [does not] include *n* banner pages at start of file
/INFORMATION	displays informational rather than fatal message for files not found and prints all others
/[NO]LOG	[does not] log file names on the console terminal as they are printed
/NAME:[dev:]jobname	specifies a QUEUE job name; with *dev*, sends files to that device
/NEWFILES	prints all files with current system date
/PRINTER	forces output to printer with PIP even if QUEUE is running
/PROMPT	valid only when QUEUE is running; allows multiple input lines, terminated with double slash (//)
/QUERY	asks if specific files should be printed
/SINCE[:date]	prints files created on or after *date*
/WAIT	invalid when QUEUE is running; waits for volume to be mounted before executing the command


PROTECT
/BEFORE[:date]
/SINCE[:date]
/DATE[:date]
/NEWFILES
/EXCLUDE
/INFORMATION
/[NO]LOG
/QUERY
/SETDATE[:date]
/SYSTEM
/WAIT
SP
filespecs

The PROTECT command protects files so that they cannot be deleted.

/BEFORE[:date]	protects only those files created before *date*
/DATE[:date]	protects only those files created on *date*
/EXCLUDE	protects all files except those you specify
/INFORMATION	displays informational rather than fatal message for files not found and protects all others
/[NO]LOG	[does not] list on the console terminal all files protected by the command
/NEWFILES	protects only files with the current system date
/QUERY	asks for confirmation before protecting each file
/SETDATE[:date]	assigns *date* to all newly protected files
/SINCE[:date]	protects only those files created on or after *date*
/SYSTEM	protects any .SYS files that are specified
/WAIT	waits for volume to be mounted before executing the command

R (SP) filespec

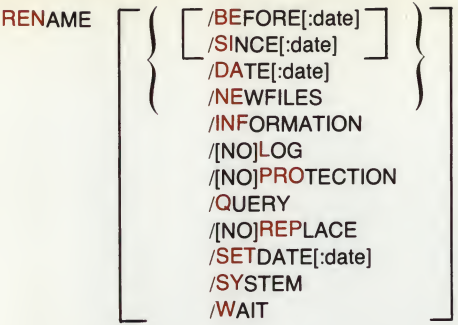
The R command runs programs from the system volume.

REENTER

The REENTER command starts a program at its reentry address.

REMOVE Ⓢ device[,...device]

The REMOVE command clears a device name from the system tables.



The RENAME command assigns a new name to an existing file.

/BEFORE[:date]	renames only files created before *date*
/DATE[:date]	renames only files created on *date*
/INFORMATION	displays informational rather than fatal message for files not found and renames all others
/[NO]LOG	[does not] list files renamed on the console terminal
/NEWFILES	renames only files with the current system date
/[NO]PROTECTION	[does not] mark a file as protected

/QUERY	asks for confirmation before renaming a file
/[NO]REPLACE	[does not] replace an existing file having the same name
/SETDATE[:date]	assigns *date* to all renamed files
/SINCE[:date]	renames files created on or after *date*
/SYSTEM	allows renaming of .SYS files
/WAIT	waits for volume to be mounted before executing the command

RESET

The RESET command initializes background system tables.

RESUME (SP) jobname

The RESUME continues execution of a foreground or system job after a SUSPEND command.

RUN (SP) filespec [{ (SP) input-list [(SP) output-list] }]
                   { (SP) argument   }

The RUN command loads and runs a program from the default device DK, or from another file-structured device.

SAVE Ⓢ filespec [Ⓢ parameters]

The SAVE command writes selected areas of memory to a file. Parameters are of the form:

address[-address][,address][-address]

SET Ⓢ physical-device-name Ⓢ condition[,...condition]
item

The SET command changes characteristics of device handlers and the values of some system parameters. Refer to the list of SET commands at the end of this section for a complete list of all SET options.

SHOW [SP] (ALL | CONFIGURATION | DEVICES[:dev] | ERRORS [/ALL | /FILE:filespec | /FROM[:date] | /OUTPUT:filespec | /PRINTER | /SUMMARY | /TERMINAL | /TO[:date]] | JOBS | MEMORY | QUEUE | SUBSET | TERMINALS)

The SHOW command lists on the console terminal information about the monitor configuration and the available hardware.

ALL	includes the CONFIGURATION, DEVICES, device assignments (SHOW with no arguments) JOBS, TERMINALS, MEMORY, and SUBSET options
CONFIGURATION	lists the monitor version, patch level, monitor SET options in effect, hardware configuration, and special features
DEVICES[:dev]	lists the available device handlers, status, and vector addresses; with *dev*, displays information about only that device
ERRORS	provides information about I/O transfers
/ALL	lists all errors that have occurred
/FILE:filespec	lists error log from a file
/FROM[:date]	lists all errors that occurred on or after *date*
/TO[:date]	lists all errors that occurred before *date*
/OUTPUT:filespec	writes the error report to a file
/PRINTER	prints error log on the line printer
/SUMMARY	lists I/O errors only
/TERMINAL	displays error log on the console terminal
JOBS	lists job names and numbers, assigned console, priority, state, and memory limits
MEMORY	lists organization of physical memory, locations of jobs and device handlers, and base addresses of the USR and KMON
QUEUE	valid only when QUEUE is running; lists contents of printer queue
SUBSET	lists logical disk subsetting currently in effect
TERMINALS	lists status of any active terminals; valid only with multiterminal support
no option	lists installed devices and device assignments

SQUEEZE [/OUTPUT:device | /[NO]QUERY | /WAIT] (SP) device

The SQUEEZE command consolidates all unused blocks and directory entries on a volume.

/OUTPUT:device	copies files from input volume to output volume specified in *device*
/[NO]QUERY	[does not] ask for confirmation before performing the squeeze operation
/WAIT	waits for volume to be mounted before executing the command

SRUN (SP) filespec [/BUFFER:n
/LEVEL:n
/NAME:logical-jobname
/PAUSE
/TERMINAL:n]

The SRUN command starts system jobs.

/BUFFER:n	allocates *n* words of memory in addition to actual program size
/LEVEL:n	assigns execution priority level, where *n* = 1–6
/NAME:jobname	assigns a logical job name to a program
/PAUSE	prints load address of program and waits for RESUME command
/TERMINAL:n	assigns terminal *n* to the system job

START [SP address]

The START command begins execution of a program at the specified address.

SUSPEND [SP jobname]

The SUSPEND command temporarily stops execution of a foreground or system job.

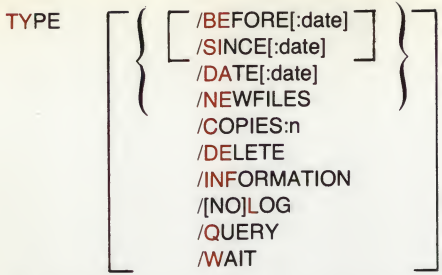
TECO filespec

The TECO command is available only under the FB and XM monitors. It is equivalent to:

```
.R TECO
*EBfilespec$$
```

TIME [SP] hh:mm:ss]

The TIME command displays or sets the system time.

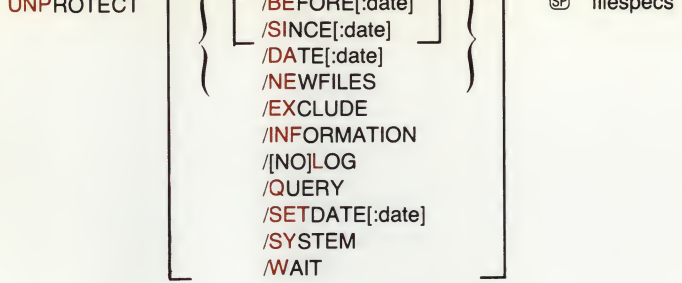


The TYPE command prints the contents of one or more files on the console terminal.

/BEFORE[:date]	prints only files created before *date*
/COPIES:n	prints *n* copies of a file; *n* can be from 2 to 32
/DATE[:date]	prints only files created on *date*
/DELETE	deletes a file after printing
/INFORMATION	displays informational rather than fatal messages for files not found and prints all others
/[NO]LOG	[does not] print names of files as they are typed
/NEWFILES	prints only files with the current system date
/QUERY	asks for confirmation before printing file
/SINCE[:date]	prints only files created on or after *date*
/WAIT	waits for volume to be mounted before executing the command

UNLOAD [device[,...device]]
[jobname[,...jobname]]

The UNLOAD command removes previously loaded handlers from memory. It also removes terminated foreground and system jobs.


UNPROTECT
/BEFORE[:date]
/SINCE[:date]
/DATE[:date]
/NEWFILES
/EXCLUDE
/INFORMATION
/[NO]LOG
/QUERY
/SETDATE[:date]
/SYSTEM
/WAIT
SP
filespecs

The UNPROTECT command removes protection from files so that they can be deleted.

/BEFORE[:date]	removes protection from files created before *date*
/DATE[:date]	removes protection from files created on *date*
/EXCLUDE	removes protection from all files except those you specify
/INFORMATION	prints informational rather than fatal messages for files not found and removes protection from all others
/[NO]LOG	[does not] list on the console terminal the names of all files unprotected by the command
/NEWFILES	removes protection from only those files that have the current system date
/QUERY	asks for confirmation before removing protection from each file
/SETDATE[:date]	assigns *date* to all newly unprotected files
/SINCE[:date]	removes protection from all files created on or after *date*
/SYSTEM	removes protection from any .SYS files that are specified
/WAIT	waits for volume to be mounted before executing the command

SET Commands

Both the monitor and the device handlers have SET commands. Most of the device handler SET commands affect only the copy of the handler on SY:. Any copy of the handler in memory is not affected. The few exceptions are marked with an asterisk (*) or a plus (+). An asterisk indicates that any memory-resident copy of the handler and the disk-resident copy of the handler are changed. A plus indicates that only the memory-resident copy of the handler is changed.

The monitor SET commands, such as SET USR NOSWAP, set or clear status words in the resident monitor.

Device or Item	Condition	Effect
CR	CODE=n	Modifies the card reader handler to use either DEC 026 or **DEC 029** card codes
	CRLF	Appends a carriage return/linefeed to each card image
	NOCRLF	Transfers each card image without appending a carriage return/linefeed
	HANG	Waits for user correction if reader is not ready
	NOHANG	Generates an immediate error if reader is not ready
	IMAGE	Reads each card as a 12-bit binary number
	NOIMAGE	Translates card codes to ASCII, as set by the CODE option
	TRIM	Removes trailing blanks
	NOTRIM	Transfers full 80 columns per card

DD	CSR=n	Uses *n* as the CSR address for the first DECtape II controller
	CSR2=n	Uses *n* as the CSR address for the second DECtape II controller
	RETRY=n	Sets the number of retries after an I/O error
	SUCCES	Logs successful I/O transfers as well as errors when running the error logger
	NOSUCCES	Logs only I/O errors
	VECTOR=n	Uses *n* as the vector address for the first DECtape II controller
	VEC2=n	Uses *n* as the vector address for the second DECtape II controller
DL	CSR=n	Uses *n* as the CSR address for the DL handler
	RETRY=n	Sets the number of retries after an I/O error
	SUCCES	Logs successful I/O transfers as well as errors when running the error logger
	NOSUCCES	Logs only I/O errors
	VECTOR=n	Uses *n* as the vector address for the DL handler
DM	CSR=n	Uses *n* as the CSR address for the DM handler
	RETRY=n	Sets the number of retries after an I/O error
	SUCCES	Logs successful I/O transfers as well as errors when running the error logger
	NOSUCCES	Logs only I/O errors
	VECTOR=n	Uses *n* as the vector address for the DM handler
DUx	CSR=n	Uses *n* as the CSR address for the first port of the DU MSCP controller

	CSRm=n	Valid only if handler is assembled for multiple ports; uses *n* as the CSR address for DU port *m*; *m* can be 2, 3, or 4
	PART=n	Assigns *DUx* to MSCP partion *n*; initial *n* is 0
	PORT=n	Valid only if handler is assembled for multiple ports; assigns *DUx* to MSCP port *n*; initial *n* is 0
	RETRY=n	Sets the number of retries after an I/O error
	SUCCES	Logs successful I/O transfers as well as errors when running the error logger
	NOSUCCES	Logs only I/O errors
	UNIT=n	Assigns *DUx* to MSCP unit *n*; initial setting defines *n* equal to *x*
	VECTOR=n	Uses *n* as the vector address for the first port of the DU MSCP controller
	VECm=n	Valid only if handler is assembled for multiple ports; uses *n* as the vector address for port *m*; *m* can be 2, 3, or 4
DXx	CSR=n	Uses *n* as the CSR address for the DX handler
	RETRY=n	Sets the number of retries after an I/O error
	SUCCES	Logs successful I/O transfers as well as errors when running the error logger
	NOSUCCES	Logs only I/O errors
	VECTOR=n	Uses *n* as the vector address for the DX handler
	* **WRITE**	Write enables DX unit *x*
	* NOWRITE	Write locks DX unit *x*

DYx	CSR=n	Uses *n* as the CSR address for the DY handler
	RETRY=n	Sets the number of retries after an I/O error
	SUCCES	Logs successful I/O transfers as well as errors when running the error logger
	NOSUCCES	Logs only I/O errors
	VECTOR=n	Uses *n* as the vector address for the DY handler
	* **WRITE**	Write enables DY unit *x*
	* NOWRITE	Write locks DY unit *x*
EDIT	**EDIT**	Defines EDIT.SAV to be the default text editor for the keyboard monitor EDIT command
	KED	Defines KED.SAV to be the default text editor; requires a VT100 terminal
	K52	Defines K52.SAV to be the default text editor; requires a VT52 terminal
	TECO	Defines TECO.SAV to be the default text editor
EL	+ LOG	Turns on error logging in SJ monitor
	+ NOLOG	Turns off error logging in SJ monitor
	+ PURGE	Clears internal error log buffer in SJ monitor
ERROR	WARNING	Aborts indirect command files and KMON commands if warnings, errors, or severe or fatal errors occur
	ERROR	Aborts indirect command files and KMON commands if errors or severe or fatal errors occur

	{ FATAL SEVERE }	Aborts indirect command files and KMON commands if severe or fatal errors occur
	NONE	Allows indirect command files and KMON commands to continue, if possible, despite severe or fatal errors
EXIT	**SWAP**	Saves running program in swap blocks when program exits so that it can be restarted with REENTER command
	NOSWAP	Disables writing of program to swap blocks when program exits
KMON	IND	Causes indirect command files (@filespec syntax) to be interpreted by IND.SAV
	NOIND	Causes indirect command files to be interpreted by KMON. The syntax $@filespec forces NOIND interpretation even when SET IND is in effect
LDx	* CLEAN	Checks all logical disk assignments to make sure they are valid
	* FREE	Deassigns logical disk unit *x*
	* **WRITE**	Write enables logical disk unit *x*
	* NOWRITE	Write locks logical disk unit *x*
LP	CR	Passes carriage return characters (octal 15) to the line printer
	NOCR	Does not pass carriage return characters to the line printer
	CSR=n	Uses *n* as the CSR address for the LP handler

	CTRL	Passes all characters, including non-printing control characters, to the line printer
	NOCTRL	Ignores non-printing control characters
	FORM	Uses the formfeed character (octal 14) to generate formfeeds
	NOFORM	Uses multiple linefeed characters to simulate formfeeds
	FORM0	Issues a formfeed before printing block 0
	NOFORM0	Does not issue a formfeed before printing block 0
	HANG	Waits for user correction when the line printer is not ready
	NOHANG	Generates an immediate error when the line printer is not ready
	LC	Sends lowercase characters to the line printer
	NOLC	Translates lowercase characters to uppercase before printing
	LENGTH=n	Defines length of page; initial value of *n* is 66
	SKIP=n	Defines number of lines to skip at page break; use with SET LP NOFORM
	TAB	Sends TAB characters (octal 11) to the line printer
	NOTAB	Simulates tabs with spaces
	VECTOR=n	Uses *n* as the vector address for the line printer
	WIDTH=n	Sets line width to *n*; *n* can be 30 to 255(decimal); initial value of *n* is 132
LS	CR	Passes carriage return characters (octal 15) to the line printer
	NOCR	Does not pass carriage return characters to the line printer

CSR=n	Uses *n* as the CSR address for the LS handler
CTRL	Passes all characters, including non-printing control characters, to the line printer
NOCTRL	Ignores non-printing control characters
FORM	Uses the formfeed character (octal 14) to generate formfeeds
NOFORM	Uses multiple linefeed characters to simulate formfeeds
FORM0	Issues a formfeed before printing block 0
NOFORM0	Does not issue a formfeed before printing block 0
HANG	Waits for user correction when the line printer is not ready
NOHANG	Generates an immediate error when the line printer is not ready; requires device timeout support in the handler
LC	Sends lowercase characters to the line printer
NOLC	Translates lowercase characters to uppercase before printing
LENGTH=n	Defines length of page; initial value of *n* is 66
SKIP=n	Defines number of lines to skip at page break; use with SET LS NOFORM
TAB	Sends TAB characters (octal 11) to the line printer
NOTAB	Simulates tabs with spaces
VECTOR=n	Uses *n* as the vector address for the LS handler
WIDTH=n	Sets line width to *n*; *n* can be 30 to 255(decimal); initial value of *n* is 132

MM	DEFALT=9	Sets defaults for 9-track magtape
	DENSE=n	Sets density for 9-track magtape handler; *n* can be **800**, 809, or 1600
	ODDPAR	Sets odd parity for 9-track magtape
	NOODDPAR	Sets even parity for 9-track magtape
MT	DEFALT=n	Sets defaults for 7- or **9**-track magtape; *n* can be 7 or 9
	DENSE=n	Sets density for 7- or 9-track magtape; *n* can be 200, 556, **800**, 807, or 809
	DUMP	Writes bytes to 7-track magtape at 800 bbpi
	ODDPAR	Sets odd parity for 7- or 9-track magtape
	NOODDPAR	Sets even parity for 7- or 9-track magtape
PDx	* **WRITE**	Write enables PD unit *x*
	* NOWRITE	Write locks PD unit *x*
RK	CSR=n	Uses *n* as the CSR address for the RK handler
	RETRY=n	Sets the number of retries after an I/O error
	SUCCES	Logs successful I/O transfers as well as errors when running the error logger
	NOSUCCES	Logs only I/O errors
	VECTOR=n	Uses *n* as the vector address of the RK handler
SL	ASK	Automatically checks for type of terminal
	KMON	Enables use of SL by KMON but not by user programs

	LEARN	Leaves help text on screen and scrolls input below it
	NOLEARN	Does not keep help text on screen
	OFF	Disables single-line editor
	ON	Enables single-line editor for use both by KMON and by user programs and loads the SL handler
	SYSGEN	Configures the SL handler to match the current monitor's SYSGEN parameters
	TTYIN	Enables single-line editing of .TTYIN input; if special made is set (bit 12 of the JSW) this command has no effect
	NOTTYIN	Disables single-line editing of .TTYIN input
	VT52 VT62 VT100 VT101 VT102	Configures the SL handler to use the type of terminal you specify
	WIDTH=n	Sets line width to *n*; *n* can be 30 to 132(decimal); initial value of *n* is 79
{ TERM TT }		
	+ CONSOL=n	Changes background console terminal to terminal defined as logical terminal *n*; requires multiterminal support
	+ **CRLF**	Automatically inserts carriage return/linefeed when you attempt to type past the right margin
	+ NOCRLF	No special action is taken at the right margin

	+ **FB**	Treats CTRL/F, CTRL/B, and CTRL/X as program control characters
	+ NOFB	Assigns no special meaning to CTRL/F, CTRL/B, and CTRL/X
	+ FORM	Indicates that the console has hardware formfeed capability
	+ **NOFORM**	Simulates formfeeds with eight linefeeds
	+ **HOLD**	Enables Hold Screen mode for VT50/VT52
	+ NOHOLD	Disables Hold Screen mode for VT50/VT52
	+ **PAGE**	Interprets CTRL/S as XOF and CTRL/Q as XON to stop and start terminal output, respectively
	+ NOPAGE	Disables special handling of CTRL/S and CTRL/Q
	+ QUIET	Disables echoing of lines from indirect command files
	+ **NOQUIET**	Enables echoing of indirect command file lines as they are processed
	+ SCOPE	Echoes RUBOUT characters as backspace/space/backspace
	+ **NOSCOPE**	Echoes RUBOUT as backslash followed by the character(s) deleted
	+ TAB	Indicates that the console terminal has hardware tab stops
	+ **NOTAB**	Simulates tab stops every eight positions
	+ WIDTH=n	Sets the terminal width to *n*; *n* can be 30 to 255(decimal); initial value of *n* is 72
USR	**SWAP**	Allows the background job to swap the USR
	NOSWAP	Locks the USR in memory so that it cannot be swapped

VM	BASE = n	VM handler uses only memory above *n*; value of *n* is physical address divided by 100(octal)
WILD	EXPLICIT	The system recognizes file specifications exactly as typed
	IMPLICIT	The system fills in missing fields in file specifications with wildcards (asterisks)

RT–11 Utility Program Options and ODT Command Summary

RT–11 utilities use the Command String Interpreter (CSI) to process the command lines typed to them.

CSI Command String Format

output-filespecs/options = input-filespecs/options

Syntax

output-filespec (up to three files)

dev:filnam.typ[n],...dev:filnam.typ[n]

input-filespec (up to six files)

dev:filnam.typ,...dev:filnam.typ

/option

/x:oval or /x:dval.

Mnemonic	Meaning
dev:	1- to 3-character device name
filnam	1- to 6-character file name
typ	0- to 3-character file type
[n]	length of output file; brackets are part of syntax
/x	option character
oval	optional octal argument or Radix–50 3-character argument
dval.	optional decimal argument; indicated by decimal point
=	separates output and input filespecs

Some utility programs, such as BINCOM, have particular meanings assigned to particular files in the CSI command string. In these cases, the syntax of the utility's command line is shown in detail.

BINCOM (Binary Compare Program)

Syntax

.R BINCOM
*[listfile][,SIPPfile] = oldfile,newfile[/options]

Options

/B	Compares bytes instead of words
/D	Compares two entire volumes
/E:n	Ends comparison at block *n*
/H	Prints help information on terminal
/O	Creates a differences listing file or SIPP command file even if no differences between the input files are found
/Q	Suppresses terminal output of differences
/S:n	Starts comparison at block *n*

BUP (Backup Utility Program)

Options

/I	Backs up a large volume on multiple smaller volumes
/L	Lists directory of a backup volume
/X	Restores a large file or, with /I, a large volume from multiple backup volumes
/Z	Initializes a volume for backup
no option	Backs up a large file on multiple smaller volumes

DIR (Directory Listing Program)

Options

/A	Lists directory alphabetically
/B	Includes starting block numbers in directory listing
/C:n	Lists directory in *n* columns; *n* can be 1 to 9
/D[:date]	Includes only files with *date*
/E	Lists entire directory, including unused spaces
/F	Lists short format directory in five columns
/G	Lists directory entry of specified file and all subsequent directory entries
/J[:date]	Lists files created on or after *date*
/K[:date]	Lists files created before *date*
/L	Lists volume directory in order of entry
/M	Lists unused areas
/N	Lists directory summary
/O	Gives sizes and block numbers in octal
/P	Lists all files except those you specify
/Q	Lists deleted files
/R	Sorts directory in reverse order; use with /S
/S[:xxx]	Sorts directory listing; *xxx* can be DAT, NAM, POS, SIZ, or TYP
/T	Lists only protected files
/U	Lists only unprotected files
/V[:ONL]	Includes volume ID and owner name as part of directory listing; with ONL, lists only ID and name

DUMP (File Dump Program)

Options

/B	Outputs octal bytes
/E:n	Ends output at block *n*
/G	Ignores input errors
/N	Suppresses ASCII output
/O:n	Outputs only block *n*
/S:n	Starts output at block *n*
/T	Defines a magtape as non-RT–11 file structured
/W	Outputs octal words
/X	Outputs Radix–50 characters

Options

/B[:RET]	Writes FILE.BAD entries over bad blocks; use with /Z; with RET, retains FILE.BAD entries created by previous initialization
/C	Creates a file; use with /G:n
/D	Restores previously initialized volume
/E:n	Specifies last block number; use with /I or /K
/F	Prints names of files with bad blocks; use with /K
/G:n	Specifies starting block number; use with /C, /I, or /K
/H	Verifies after copying; use with /I
/I	Copies image of one volume to another
/K	Scans a volume for bad blocks
/N:n	Defines number of directory segments; use with /Z; *n* can be 1 to 37(octal)
/O	Boots a volume or file
/Q	Boots a volume that is not RT V4 or later; use with /O
/R[:RET]	Scans volume for bad blocks and creates a block replacement table; with *RET*, retains previous table
/S	Consolidates free space on a volume
/T:n	Extends a file by *n* blocks; *n* free blocks must follow the file
/U[:dev]	Writes bootstrap into blocks 0, 2 through 5 of a volume; *dev* is the target device name
/V[:ONL]	Prints user ID and owner name; use with /Z to write new directory, ID, and name on volume; with ONL, writes only a new ID and name
/W	Waits for volume to be mounted before executing the command
/X	Prevents automatic reboot after using /S on system device
/Y	Suppresses query messages
/Z[:n]	Initializes device directory; *n* is the number of extra words in each directory entry

ERROUT (Error Log Report Writer)

Options

/A	Prints report on each error and a summary report on all errors and I/O transfers
/F:date	Prints report of all errors that occurred on or after the date you specify
/S	Prints summary report only
/T:date	Prints report of all errors that occurred before the date you specify

FILEX (File Exchange Program)

Options

/A	Transfers ASCII files character-by-character, deleting, nulls and rubouts; CTRL/z is interpreted as EOF
/D	Deletes a file; valid for DOS/BATCH, RSTS DECtape, interchange diskette
/F	Lists short directory
/I	Transfers in image mode
/L	Lists full directory
/P	Transfers in packed image mode
/S	Indicates volume is DOS/BATCH or RSTS format
/T	Indicates volume is DECsystem–10 DECtape
/U[:n]	Indicates volume is interchange diskette; *n* defines size of output record; default *n* is 80
/V[:ONL]	Writes a volume identification on interchange diskette; use with /U/Z; with ONL, writes only volume ID and keeps old directory
/W	Waits for volume to be mounted before executing the command
/Y	Suppresses query messages
/Z	Initializes device directory; valid for DOS/BATCH, RSTS DECtape, interchange diskette

FORMAT (Volume Formatting Program)

FORMAT can format RK05, RX01, RX02, and RK06–RK07 disks.

Options

/P[:n] Defines patterns to be used during verification pass.

n	pattern	n	pattern
1	000000	100	021042
2	111111	**200**	104210
4	163126	400	155555
10	125252	1000	145454
20	052525	2000	146314
40	007417		

/S Formats diskettes in single-density format

/V[:ONL] Formats, then verifies volume; with ONL, verifies without formatting

/W Waits for volume to be mounted before executing the command

/Y Suppresses query messages

no option Formats without verification, assumes diskettes are double-density

Directive

Label Definition

.label:

Assigns the name represented by *label* to a line in the control file so that the line can be referenced.

Symbol Definition

.ASK [def:time] symbol prompt

Defines or redefines a logical symbol and assigns the symbol a logical (true or false) value based on a response to a prompt.

.ASKN [low:high:def:time] symbol prompt

Defines or redefines a numeric symbol and assigns it a numeric value based on a response to a prompt.

.ASKS [low:high:"def":time] symbol prompt

Defines or redefines a string symbol and assigns it a string value based on a response to a prompt.

.DUMP symboltable

Displays local, global, and special symbol definitions.

.ERASE LOCAL [symbolname]

.ERASE GLOBAL [symbolname]

Deletes local or global symbol definitions.

.PARSE string "control-string" sym1 sym2 ....symn

Breaks a string into substrings, which IND stores in string symbols.

.SETD numericsymbol

.SETO numericsymbol

Redefines a numeric symbol to a decimal (.SETD) or octal (.SETO) radix.

.SETL logicalsymbol logicalexpression

Defines or redefines a logical symbol and assigns it a logical value.

.SETN numericalsymbol numericalexpression

Defines or redefines a numeric symbol and assigns it a numeric value.

.SETS stringsymbol stringexpression

Defines or redefines a string symbol and assigns it a string value.

.SETT logicalsymbol

.SETT [mask] numericsymbol

.SETF logicalsymbol

.SETF [mask] numericsymbol

Defines or redefines a logical symbol or redefines bits within a numeric symbol, and assigns the symbol or bits a true or false value.

.VOL stringsymbol device

Assigns the volume identification of a volume to a string symbol.

File Access

.CHAIN filespec[/options]

Closes the current control file, opens another control file, and resumes execution.

.CLOSE [#n] [filespec]

Closes an output data file.

.DATA [#n] text-string

Specifies a single line of data to be sent to an output data file.

.OPEN [#n] filespec

Creates an output data file. If the file you specify with .OPEN already exists, .OPEN creates a new tentative file with the same name. If you subsequently use the .CLOSE directive, .CLOSE will delete the old file and make the new file permanent. Use the .OPEN directive only when you wish to write to a file.

.OPENA [#n] filespec

Opens an existing file and appends subsequent data to it. If the file you specify does not already exist, .OPENA creates a new file. Use this directive only when you wish to write to a file.

.OPENR [#n] filespec

Opens an existing file for use with the .READ directive. Use this directive only when you wish to read from a file.

.PURGE [#n]

Deactivates an output file, resulting in no changes to already existing files.

.READ [#n] stringsymbol

Reads the next record from a file into a string variable. The file must have been previously opened with .OPENR.

.TESTFILE filespec

Determines whether or not a file exists.

Logical Control

.BEGIN

Marks the beginning of a Begin-End block.

.END

Marks the end of a Begin-End block.

.EXIT [value]

Terminates processing of either a Begin-End block or the current control file and returns control to the previous level; can also assign a value to the numeric symbol <EXSTAT>.

.GOSUB label

Branches to a subroutine within the control file.

.GOTO label

Branches to another location in the control file.

.ONERR label

Upon detecting an error, branches to another location in the control file.

.RETURN

Returns control from a subroutine to the line immediately following that subroutine's call.

.STOP

Terminates control file processing.

Logical Tests

.IF symbol operator expression action

Determines whether or not a symbol satisfies one of several possible conditions.

.IFDF symbol action

.IFNDF symbol action

Determines whether or not a symbol is defined.

.IFENABLED opmode action

.IFDISABLED opmode action

Determines whether an operating mode is enabled or disabled.

.IFLOA device action

.IFNLOA device action

Determines whether or not a device handler has been loaded.

.IFT logicalsymbol action

.IFT [mask] numericsymbol action

.IFF logicalsymbol action

.IFF [mask] numericsymbol action

Determines whether a logical symbol is true or false, or tests specific bits in a numeric symbol.

.TESTDEVICE devicename[:]

Returns information about a device in the string <EXSTRI>.

.TEST symbol

.TEST stringsymbol matchstring

Tests a symbol or string.

Execution Control

.DELAY time

Delays control file processing for a specified period of time.

Disable/Enable Operating Modes

.DISABLE mode1[,mode2,mode3...]
.ENABLE mode1[,mode2,mode3...]

Disables or enables the operating modes, which are:

DATA	LOWERCASE	SUBSTITUTION
DCL	MCR	SUFFIX
DELETE	OCTAL	TIMEOUT
ESCAPE	PREFIX	TRACE
GLOBAL	QUIET	

Increment/Decrement Numeric Symbols

.DEC numericsymbol

Subtracts 1 from the value of a numeric symbol.

.INC numericsymbol

Adds 1 to the value of a numeric symbol.

Options

/D	Deletes control file after processing
/N	Suppresses execution of KMON commands within a control file
/T	Lists each IND command line as it is executed
/Q	Suppresses display of keyboard commands and responses

KED (Keypad Editor)

Options

/A:n	Allocates *n* blocks for output file
/C	Creates a new file
/I	Inspects input file, does not allow changes

LD.SYS (Logical Disk Handler)

LD.SYS operates in two ways: either it can be used by programs as a handler to access logical disks, or it can be run as a program to initialize logical disk parameters.

Options

/A:ddd	Assigns logical name *ddd* to a logical disk unit; use with /L:n
/C	Verifies all logical disk assignments; must be alone on command line
/L:n	Assigns or deassigns logical disk unit *n*
/R:n	Write locks logical disk unit *n*
/W:n	Write enables logical disk unit *n*

LIBR (Librarian)

Syntax

.R LIBR
*[libraryfile][,listfile] = inputfiles[/options]

Options

/A	Includes in library directory all global symbols including absolute global symbols
/C	Allows multiple input lines
/D	Deletes a module from a library file
/E	Extracts a module from a library and stores it as a .OBJ file
/G	Deletes a global symbol from a library directory
/M:n	Creates a macro library from an ASCII input file and allocates *n* blocks for the macro name directory
/N	Includes module names in library directory
/P	Includes psect names in library directory
/R	Replaces modules in a library file
/U	Updates (inserts and replaces) modules in a library file
/W	Produces a wide (132 column) library directory listing
/X	Creates a library with multiple global definitions
//	Allows multiple input lines until next occurrence of //
no option	Assumes module insertion

LINK (Linker)

Syntax

.R LINK
*[binfile][,mapfile][,stbfile] = objfiles[/options]

Options

/A	Lists global symbols in alphabetical order
/B:n	Sets bottom address of program to *n*; invalid with /H and /R
/C	Continues input on new line; do not use with //
/D	Allows duplicate library subroutines
/E:n	Extends root program segment to specific value
/F	Uses default FORTRAN library FORLIB.OBJ when linking
/G	Increases size of linker's library directory buffer
/H:n	Specifies highest address to be used by relocatable code; invalid with /B, /Q, /R, /Y
/I	Extracts specified global symbols from a library
/K:n	Inserts value of *n* into word 56 of block 0 as virtual .SETTOP high limit; *n* can be 1 to 32(decimal); valid only with /V
/L	Produces output file in LDA format; invalid with /R, /V
/M[:n]	Defines stack address
/N	Produces global cross-reference listing as part of load map
/O:n	Defines overlay structure; invalid with /L
/P:n	Changes amount of space linker uses for library routines list; default is 170
/Q	Specifies start addresses of up to eight root program sections; invalid with /R, /H
/R[:n]	Produces output in .REL format; *n* is the stack size; invalid with /B, /H, /K, /L, Q

/S	Allows maximum memory space for linker symbol table
/T[:n]	Defines transfer address
/U:n	Rounds up a program section size; *n* must be a power of 2
/V	Enables special XM monitor .SETTOP and .LIMIT features; invalid with /L
/W	Produces a wide load map listing (132 columns)
/X	Does not output bitmap if code is below 400
/Y:n	Starts a program section on address boundary *n*; invalid with /H
/Z:n	Sets unused locations to *n*
//	Allows multiple input lines; use on first and last lines of input

MACRO (Macro Assembler)

Syntax

.R MACRO
*[objfile][,listfile][,creffile] = inputfiles[/options]

Temporary files for MACRO and CREF are normally assigned to DK. Assign the logical name WF to reassign MACRO's work file; assign the logical name CF to reassign CREF's temporary file.

Options

/C[:arg] Generates cross-reference table; *arg* can be
- C — Control and program sections
- E — Errors
- M — Macro names
- P — Permanent symbols
- R — Registers
- S — User-defined symbols
- **no argument** — equivalent to /C:E:M:S

/D:arg Specifies .DSABL directives; *arg* can be
- ABS — Produces absolute binary output
- AMA — Assembles all relative addresses as absolute addresses
- CDR — Treats source columns beyond column 72 as comments
- DBG — Outputs ISD records as part of .OBJ file
- FPT — Floating-point truncation
- GBL — Treats undefined symbols as globals
- LC — Accepts lowercase ASCII input
- LSB — Local symbol block
- MCL — Automatic .MCALL
- PNC — Binary output
- REG — Defines default register mnemonics

/E:arg Specifies .ENABL directives; refer to values for *arg* under /D

/L:arg — Enables listing directives; *arg* can be

- BEX — Binary expansions of ASCII text
- BIN — Generated binary code
- CND — Unsatisfied conditionals and all .IF and .ENDC statements
- COM — Comments
- LD — Listing directives having no arguments
- LOC — Location counter
- MC — Macro calls and repeat range expansions
- MD — Macro definitions and repeat range expansions
- ME — Macro expansions
- MEB — Macro expansion binary code
- SEQ — Line numbers
- SRC — Source code
- SYM — Symbol table
- TOC — Table of contents
- TTM — Listing output format in 80-column format; default is 132-column format

/M — Macro library input file

/N:arg — Disables listing directives; refer to values for *arg* under /L

ODT/VDT (Octal Debugger)

Commands

RET	Closes open location and accepts next command
LF	Closes current location and opens next sequential location
^	Opens previous location
_	Indexes contents of opened location by PC and opens resulting location
>	Uses contents of opened location as relative branch and opens that location
<	Returns to sequence prior to @, >, or _ and opens next location
@	Uses contents of opened location as absolute address and opens that location
r/	Opens word at location *r*
/	Reopens last opened location
r\	Opens byte at location *r*
\	Reopens last opened location as byte
I	Prints address of opened location relative to the closest relocation register
nI	Prints address of opened location relative to relocation register *n*
$n/	Opens general register *n*
$B/	Opens first word of breakpoint table
$C/	Opens constant register
$F/	Opens format register
$M/	Opens first mask register
$P/	Opens priority register
$R/	Opens first relocation register
$S/	Opens status register
r;nA	Prints *n* bytes of ASCII starting at location *r*, then allows *n* bytes of input from the terminal
;B	Removes all breakpoints
r;B	Sets next free breakpoint at location *r*
r;nB	Sets breakpoint *n* at location *r*
;nB	Removes breakpoint *n*
n;C	Stores *n* in constant register

r;E	Searches for instructions that reference effective address *r*
;F	Fills memory with the contents of the constant register
r;G	Starts execution of program at location *r*
;I	Fills memory bytes with low 8 bits of the constant register
r;O	Calculates offset from current location to location *r*
;P	Execution proceeds from breakpoint
n;P	Execution proceeds from breakpoint; stops after encountering the breakpoint *n* times
;R	Sets all relocation registers to 177777
;nR	Sets relocation register *n* to 177777
m;nR	Sets relocation register *n* to value *m*; default *n* is 0
R	Subtracts contents of closest relocation register less than or equal to the contents of the current location from the contents of the location, and prints result
nR	Subtracts contents of relocation register *n* from contents of current location and prints result
!	Calculates address of the current location relative to the closest relocation register less than or equal to the address of the current location, and prints result
n!	Calculates address of the current location relative to relocation register *n*, and prints result
;S	Disables single-step mode
;nS	Enables single-step mode; *n* can be any integer
n;W	Searches for words matching value *n*
X	Prints contents of opened location as Radix–50; accepts three Radix–50 characters as input

PAT (Object Module Patch Program)

Options

/C[:n]	Calculates [or verifies] module checksum

PIP (Peripheral Interchange Program)

Options

/A	Copies files in ASCII mode
/B	Copies files in formatted binary mode
/C[:date]	Includes only files with *date*
/D	Deletes files
/E	Waits for volume to be mounted before executing the command
/F	Marks output files as protected
/G	Ignores any input errors
/H	Copies files from one large volume to several small volumes
/I[:date]	Copies only files created on or after *date*
/J[:date]	Copies files created before *date*
/K:n	Transfers *n* copies of the output files to a sequential device, such as LP:, PC:, or TT:
/M:n	Controls positioning and rewinding of magtape; refer to PIP — Use of /M:n, below
/N	Does not copy or rename a file if a file of the same name exists on the output device
/O	Deletes a file on the output device before copying, if a file of a given name already exists
/P	Copies or deletes all files not specified
/Q	Queries before performing operation
/R	Renames input filename to output filename
/S	Copies files one block at a time
/T[:date]	Puts *date* on all files copied or renamed
/U	Copies and concatenates specified files
/V	Verifies output file; do not use with /A or /B
/W	Logs operations on console
/X	Prints informational rather than fatal messages for files not found, and processes all others
/Y	Includes .SYS files in operation
/Z	Removes protection from output files
no option	Copies files in Image mode

Options /D, /F, /K, N, /O, /R, /T, /Z are invalid or restricted with magtape.

PIP — Use of /M:n

Effect when reading from magtapes

n = 0	The magtape rewinds and PIP searches for the file. If more than one name is given, PIP rewinds for each file. If a wildcard is used, PIP rewinds once and then copies all matching files as they are encountered.
n = +n	PIP goes to file with sequence number *n*. If the file at that position matches specified name, the file is copied; otherwise PIP returns *file not found* error. If a wildcard is used, PIP begins the search at file *n*.
n = –1	PIP begins the search at current position without rewinding.

Effect when writing to magtapes

n = 0	The tape rewinds before each file is copied and PIP searches for a duplicate file name. If a file of the same name is found, PIP prints a warning and does not copy the file. Otherwise, the file is written at LEOT.
n = +n	PIP goes to file sequence number *n* and enters specified file. If LEOT is encountered before file *n*, an error is printed. If more than one file name is specified or a wildcard is used, the tape does not rewind and PIP does not check for duplicate names.
n = –1	PIP goes to LEOT and enters specified file without checking for duplicate names.
n = –2	The tape rewinds before each copy operation. PIP enters the file at LEOT or at the occurrence of a duplicate file name.

QUEMAN (Queue Manager)

Options

/A	Halts QUEUE
/C[:date]	Prints only files created on *date*
/D	Deletes files after printing
/H:n	Prints *n* banner pages
/I[:date]	Prints only files created on or after *date*
/J[:date]	Prints files created before *date*
/K:n	Prints *n* copies of a file
/L	Lists current contents of the queue
/M	Deletes a file from the queue
/P	Sets QUEUE defaults for /H and deletion of work file
/Q	Asks for confirmation before printing
/R	Resumes or restarts current job
/S	Suspends current job; resume with /R
/W	Logs all files printed
/X	Prints error messages for files not found and processes all others
//	Allows multiple input command lines

RESORC (Resource Program)

Options

/A	Combines all options except /Z
/C	Identifies system device and lists monitor SET options in effect
/D	Lists available device handlers and their vectors
/H	Lists system hardware configuration
/J	Lists currently loaded jobs
/L	Lists device assignments
/M	Lists monitor type, version number, and patch level
/O	Lists special features included by SYSGEN
/S	Lists disk subsetting in effect
/T	Lists status and options in effect for active terminals on a multiterminal system
/X	Lists organization of physical memory
/Z	Combines /C, /H, /J, /M, /O

SIPP (Save Image Patch Program)

Syntax

.R SIPP
*[commandfile] = inputfile[/options]

Options

/A	Prevents automatic update of location 50, window definition block, overlay handler, or overlay tables
/C	Asks for checksum at completion of the patch
/D	Prints checksum at completion of the patch
/L	Creates command file but does not modify the input file

Commands

(RET)	Closes current location, opens and displays next location
(LF)	Closes current location, opens and displays next location
n(RET)	Deposits value *n* in current location, closes current location, opens and displays next location
/	Advances in word mode
/(RET)	Reopens current location as a word
\(RET)	Reopens current location as a byte
\	Advances in byte mode
^(RET)	Closes current location, opens and displays previous location
n^(RET)	Deposits value *n* in current location, closes current location, opens and displays previous location
;A	Interprets location contents as ASCII
;O	Gives values in octal, accepts input in octal
;Ax	Deposits ASCII character *x* in low byte of current location, opens and displays high byte in octal, accepts further input in ASCII

;R	Interprets location contents as Radix–50 characters
;Ryyy	Deposits Radix–50 characters *yyy* in current location, closes current location, opens next location and gives value in octal, accepts further Radix–50 character input
;S	Prompts for value for which to search
;V	Prints all previously entered modifications
CTRL/Y	Prompts for checksum, if /C used, and installs patch
CTRL/Z	Backs up to previous SIPP prompt

SLP (Source Language Patch Program)

Syntax

.R
*[patchedfile][,listfile] = inputfile[,commandfile][/options]

Options

/A	Disables audit trail generation
/B	Uses spaces instead of tabs to position audit trail text
/C	Computes checksum for file
/C:n	Validates checksum for file
/D	Creates a double-spaced listing
/L:n	Defines maximum length of a source line, in characters
/N	Suppresses creation of backup file
/P:n	Starts audit trail in column *n*
/S:n	Defines length of audit trail; maximum value for *n* is 16(decimal)
/T	Retains trailing tabs and blanks in updated file

Command File Operators

_	Indicates start of an update
\	Disables audit trail
%	Enables audit trail
/	Indicates the end of an update or series of updates
//	Separates concatenated SLP command files
<	Indicates that the next character should be taken as source input, not interpreted as a command; use in front of _, \, %, /, and < when they must be inserted in a file

SRCCOM (Source Compare Program)

Syntax

.R SRCCOM
*[listfile][,SLPfile] = oldfile,newfile[/options]

Options

/A	Specifies audit trail for SLP command file
/B	Compares blank lines
/C	Ignores comments and spacing
/D	Copies newfile to listfile and inserts change bars and bullets at left margin of listfile to mark differences
/F	Includes formfeeds in output file
/L:n	Defines number of lines that must agree to constitute a match; *n* can be 1 to 310; default is 3
/S	Ignores spaces and tabs when comparing
/T	Includes trailing spaces and tabs in comparison
/V:i:d	Defines characters to use as markers for inserts and deletions in place of change bars and bullets; use with /D; *i* and *d* are the numeric values for the ASCII characters to use.

Programmed Requests

RT–11 EMT Codes

EMT	Interpretation
0–337	Version 1 programmed requests with arguments both on the stack and in R0.
340–357	Programmed request with the arguments on the stack and/or in R0.
360–373	Used internally by the RT–11 monitor.
374	Programmed request with one argument; R0 contains a function code in the left byte and a channel code in the right byte.
375	Programmed request with several arguments; R0 points to a block of arguments.
376	Used internally by the RT–11 monitor.
377	Reserved; RT–11 ignores this EMT and returns control to the user program immediately.

.ABTIO EMT 374, CODE 13

This request aborts all outstanding I/O requests on an I/O channel. The .ABTIO request cannot be issued from a completion routine.

Call: .ABTIO chan

chan is the channel number for which to abort I/O requests.

Format: R0 =

13	chan

Errors:

None

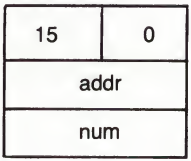
.CDFN EMT 375, Code 15

This request redefines the number of I/O channels. It allows the number of channels to be expanded from 16 to a maximum of 255(decimal) channels, numbered from 0 to 254(decimal) or 0 to 376(octal).

Call: .CDFN area, addr, num

area	is the address of a 3-word EMT argument block.
addr	is the address where the I/O channels begin.
num	is the number of I/O channels to be created.

Format: R0 –>



Errors:

Code	Explanation
0	An attempt was made to define fewer channels than already exist.

.CHAIN EMT 374, Code 10

This request allows a background program to pass control directly to another background program without operator intervention. During a .CHAIN, the monitor reverts to the original 16(decimal) channels and all nonresident device handlers are released. The area in low memory from 500 to 507 must contain the device name and file name (in Radix–50) to be chained to. The area from locations 510 to 777 is used to pass information between the chained programs.

.CHAIN cannot be used with virtual jobs.

Call: .CHAIN

Format: R0 =

10	0

Errors:

None

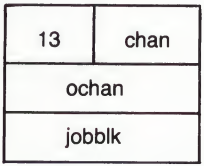
.CHCOPY EMT 375, Code 13 (FB and XM Only)

This request opens a channel, logically connecting it to a file that is currently open by another job for either input or output. A foreground, background, or system job can use this request. It must be issued before the first .READ or .WRITE request.

Call: .CHCOPY area, chan, ochan [, jobblk]

area	is the address of a 3-word EMT argument block.
chan	is the channel the current job will use to read the data.
ochan	is the channel number of the other job's channel to be copied.
jobblk	is a pointer to a 3-word ASCII logical job name that represents a system job.

Format: R0 –>



Errors:

Code	Explanation
0	Other job does not exist, does not have enough channels defined, or does not have the specified channel (ochan) open.
1	Channel (chan) already open.

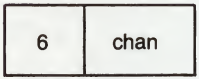
.CLOSE EMT 374, Code 6

This request terminates activity on a channel and frees it for use in another operation. The handler for the associated device must be in memory if the channel was opened with a .ENTER request.

Call: .CLOSE chan

chan is a channel number in the range 0 to 376(octal).

Format: R0 =



Errors:

Code	Explanation
3	A protected file with the same name already exists on the volume. The .CLOSE is performed anyway, resulting in two files with the same name on the volume.

.CMKT EMT 375, Code 23 (FB and XM; SJ Option)

This request cancels one or more outstanding mark time requests. The .CMKT request is a special feature in the SJ monitor.

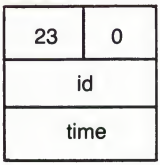
Call: .CMKT area, id [, time]

area is the address of a 3-word EMT argument block.

id is a number that identifies the mark time request to be canceled. If more than one mark time request has the same *id*, .CMKT cancels the request with earliest expiration time. If *id* is 0, .CKMT cancels all non-system mark time requests for the issuing job.

time is the address of a 2-word area in which the monitor returns the amount of time, in clock ticks, remaining in the canceled request. The first word contains the high-order time, the second contains the low-order. If an address of 0 is specified, no value is returned. If *id* is 0, the *time* parameter is ignored and need not be included.

Format: R0 –>



Errors:

Code	Explanation
0	The *id* was not 0 and a mark time request with a matching identification number could not be found.

.CNTXSW EMT 375, Code 33 (FB and XM Only)

A context switch is required when a transition is made from running one job to running another. The .CNTXSW request is used to specify locations to be included in a list when the monitor switches between background, foreground, and system jobs. This request is ignored under SJ and in XM virtual jobs.

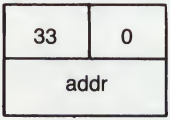
Call: .CNTXSW area, addr

area is the address of a 2-word EMT argument block.

addr is a pointer to a list of addresses terminated by a 0 word. The addresses in the list must be even and be:

a. in the range 2 to 476, or
b. in the user job area, or
c. in the I/O page (160000 to 177776).

Format: R0 –>



Errors:

Code	Explanation
0	One or more of the requirements for *addr* was violated.

.CRAW EMT 375, Code 36 (XM Only)

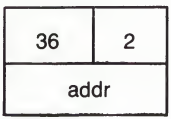
This request defines a virtual address window and optionally maps it into a physical memory region. Mapping occurs if the WS.MAP bit in the last word of the window definition block is set before .CRAW is issued.

Call: .CRAW area [, addr]

area is the address of a 2-word EMT argument block.

addr is the address of the window definition block. This argument is optional if the address pointer is already in the second word of the *area* argument block.

Format: R0 –>



Errors:

Code	Explanation
0	Window alignment error: the new window overlaps the static window for a virtual job. The window is too large or W.NAPR is greater than 7.
1	An attempt was made to define more than seven windows in the program. Eliminate a window first by using the .ELAW request, or redefine the virtual address space into fewer windows.
2	An invalid region identifier was specified.
4	The combination of the offset into the region and the size of the window to be mapped into the region is invalid.

.CRRG EMT 375, Code 36 (XM Only)

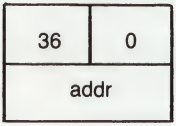
This request allocates a dynamic region in physical memory for use by the requesting job.

Call: .CRRG area [, addr]

area is the address of a 2-word EMT argument block.

addr is the address of the region definition block for the region to be created.

Format: R0 –>



Errors:

Code	Explanation
6	No region control blocks are available. Eliminate a region to obtain a region control block by using the .ELRG request, or redefine physical address space into fewer regions.
7	A region of the requested size cannot be created because not enough memory is available. The size of the largest available region is returned in R0.
10	An invalid region size was specified. Only values greater than 0 and less than or equal to 96K words are valid.

.CSIGEN EMT 344

This request calls the Command String Interpreter (CSI) in general mode to process a standard RT–11 command string. In general mode, file .LOOKUP and .ENTER requests as well as handler .FETCH requests are performed. When called in general mode, the CSI closes channels 0 to 10(octal).

In a foreground/background environment, calling the CSI performs a temporary and implicit .UNLOCK while the command line is being read.

Call: .CSIGEN devspc, defext [, cstrng [, linbuf]]

devspc	is the address of the memory area where any device handlers are to be loaded.
defext	is the address of a 4-word block that contains the Radix–50 default file types.
cstrng	is the address of the ASCIZ command string or #0 if input is from the console terminal.
linbuf	is the storage address of the original command string. This is a user-supplied area, 81 bytes long. The command string in the buffer is terminated with a 0 byte.

Errors:

Code	Explanation
0	Command line is invalid.
1	Device cannot be found in system tables.
2	Protected file of the same name already exists. A new file was not opened.
3	Volume does not have enough room to open output file(s).
4	Input file was not found.

On return, R0 points to the first available location above the handlers, the stack contains the option information, and the specified files have been opened.

.CSISPC EMT 345

This request calls the Command String Interpreter (CSI) in special mode to parse a standard RT–11 command string and return file descriptors and options to the program. In this mode, the CSI does not perform any .CLOSE, .ENTER, .LOOKUP, or handler .FETCH requests.

In a foreground/background environment, calling the CSI performs a temporary and implicit .UNLOCK while the command line is being read.

Call: .CSISPC outspc, defext [, cstrng [, linbuf]]

outspc is the address of the 39-word block to contain the file descriptors produced by .CSISPC. This area can overlay the space allocated to *cstrng*, if desired. The file descriptors are stored in Radix–50 format. The first 15 words are reserved for up to three output file descriptors. Each output file descriptor contains one word for the device name, two words for the file name, one word for the file type, and one word for any size specification, which is stored as a binary integer. The next 24 words are reserved for up to six input file descriptors. Each input file descriptor contains one word for the device name, two words for the file name, and one word for the file type.

defext is the address of a 4-word block that contains the Radix–50 default file types.

cstrng is the address of the ASCIZ input string or a 0 if input is to come from the console terminal.

linbuf is the storage address of the original command string. This is a user-specified area, 81 bytes long. The command string is terminated with a 0 byte.

Errors:

Errors are the same as in general mode except that invalid device specifications are checked only for output file specifications with null file names. Since .LOOKUP and .ENTER requests are not done, the possible error codes are:

Code	Explanation
0	Command line is invalid.
1	Device specified is not available.

CSI Option Information

In both general and special modes of the CSI, options and their associated values are returned on the stack. A CSI option is a slash (/) followed by any character. The CSI does not restrict the option to printing characters, although they should be used for clarity. The option can be followed by a value, which is indicated by a colon (:) separator. The colon separator can be followed by an octal number, a decimal number, or by one to three alphanumeric characters, the first of which must be alphabetic. Decimal values are indicated by a decimal point following the number (14.). If no decimal point is present, the number is assumed to be octal. Options can be associated with files. The format of the stack output of the CSI for options is as follows:

Word Number	Meaning
1	Number of options found in command string. If 0, no options were found
2	Even byte = 7-bit ASCII option character. Bits 8–14 = number (0–10) of the file with which the option is associated. Bit 15 = 1 if option had a value, 0 if option had no value.
3	If bit 15 of word 2 is set, word 3 contains the option value. If bit 15 is not set, word 3 contains the next option character and file number, if any.

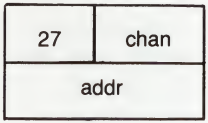
.CSTAT EMT 375, Code 27

This request returns information about a channel.

Call: .CSTAT area, chan, addr

area	is the address of a 2-word EMT argument block.
chan	is the number of the channel about which information is desired.
addr	is the address of a 6-word block to contain the status.

Format: R0 –>



Errors:

Code	Explanation
0	Channel is not open.

Information Returned:

Offset	Contents
0	Channel status word (CSW)
2	Starting block number of file
4	Length of file
6	Highest relative block number written
10	Device unit number
12	Device name in Radix–50

Channel Status Word (CSW) Bit Definitions:

HDERR$	1	Hard error
INDX$M	76	Index mask into $PNAME and other device tables
RENAM$	100	Rename operation in progress
DWRIT$	200	File opened with .ENTER; monitor will modify directory when file is closed
DBLK$M	17400	Mask for directory segment containing this entry
EOF$	20000	End-of-file found on this channel
	(14)	Reserved
ACTIV$	100000	Channel is active

.CTIMIO Macro Expansion

Use this request when writing a device handler. The .CTIMIO request cancels a pending device time-out (.TIMIO) request in the handler interrupt service section. It is used when an interrupt occurs to disable a pending .TIMIO completion routine.

Call: .CTIMIO tblk

tblk is the address of the 7-word timer block used by the .TIMIO request.

Errors:

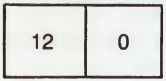
None

.DATE EMT 374

This request returns in R0 the current date information from the system date word. The year is in bits 0 to 4, the day in bits 5 to 9, and the month in bits 10 to 13. The year is the actual year minus 1972.

Call: .DATE

Format: R0 =



Errors:

None

On return, a value of 0 in R0 indicates that the system date has not been set.

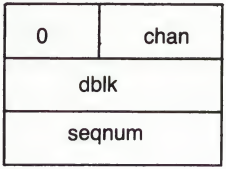
.DELETE EMT 375, Code 0

This request deletes a named file from an RT–11 volume. The channel specified must not be open. The .DELETE request is invalid for magtapes.

Call: .DELETE area, chan, dblk, [, seqnum]

area	is the address of a 3-word EMT argument block.
chan	is a free channel number in the range 0 to 376(octal).
dblk	is the address of a 4-word Radix–50 descriptor of the file to be deleted.
seqnum	file number for cassette operations; if this argument is blank, a value of 0 is assumed.

Format: R0 –>



Errors:

Code	Explanation
0	Channel is active.
1	File was not found in the volume's directory.
2	Operation is invalid; device is not file structured.
3	The file is protected and cannot be deleted.

.DEVICE EMT 375, Code 14 (FB and XM Only)

This request allows a user program to load device registers when the program terminates. The .DEVICE request sets up a list of addresses with specified values. When the job terminates, the monitor goes through the list and the designated addresses are loaded with the corresponding values.

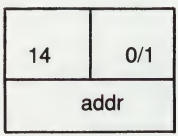
Call: .DEVICE area, addr [, link]

area is the address of a 2-word EMT argument block.

addr is the address of a list of 2-word elements, each composed of a 1-word address and a 1-word value to be put at that address. If *addr* is #0, any previous list is discarded; in this form, the argument *link* must be omitted.

link is an optional argument that, if present, specifies linking of tables on successive calls to .DEVICE.

Format: R0 –>



Errors:

None

.DRAST Macro Expansion

Use this request when writing a device handler. The .DRAST request sets up the interrupt and abort entry points, lowers the processor priority, and references a global symbol $INPTR, which contains a pointer to the INTEN routine in the resident monitor. This pointer is filled in for a system handler when the system is bootstrapped. The pointer is filled in for other handlers when they are loaded by the .FETCH request.

Call: .DRAST name, pri [, abort]

name	is the 2-character device name.
pri	is the priority of the device, and also the priority at which the interrupt service code is to execute.
abort	is an optional argument that represents the label of an abort entry point. If you omit this argument, the macro generates an RTS PC instruction at the abort entry point, which is the word immediately preceding the interrupt entry point.

This request sets up the information in block 0 and the first five words of a device handler. This macro also generates the appropriate global symbols for the handler. Before using .DRBEG, use .DRDEF to define ddCSR, ddVEC, ddDSIZ, and ddSTS.

Call: .DRBEG name

name is the 2-character device name.

The first five words of a handler are:

ddSTRT	Device vector
–	Offset to interrupt routine
–	Priority (340)
ddLQE	Pointer to last queue element
ddCQE	Pointer to current queue element

.DRBOT Macro Expansion

This macro sets up a primary driver. A primary driver must be added to a standard handler to create a system device handler. The .DRBOT macro uses the .DREND macro to mark the end of the handler so that the primary driver is not loaded into memory during normal operations.

Call: .DRBOT name, entry, read [,CONTROL = type[,...type]][,SIDES = n]

name	is the 2-character device name.
entry	is the entry point of the software bootstrap routine.
read	is the entry point of the bootstrap read routine.
type	is the type of controller supported by the handler. If CONTROL is omitted, .DRBOT defaults to UBUS and QBUS. This default is correct for all supported Version 5 handlers.
n	specifies single- or double-sided diskettes. If SIDES is omitted, .DRBOT defaults to single-sided diskettes. This default is correct for all supported Version 5 handlers.

.DRDEF Macro Expansion

This macro sets up device handler parameters, calls other device handler macros from the library, and defines symbols.

Call: .DRDEF name, code, stat, size, csr, vec

name is the 2-character device name.

code is the numeric code that is the device identifier value for the device.

stat is the device status bit pattern. The value for *stat* can use the following symbols:

VARSZ$	=	400	Handler returns volume sizes
ABTIO$	=	1000	Enter at abort entry point when job exits
SPFUN$	=	2000	Accepts .SPFUN requests
HNDLR$	=	4000	Enter at abort entry point on job abort
SPECL$	=	10000	No directory
WONLY$	=	20000	Write-only
RONLY$	=	40000	Read-only
FILST$	=	100000	File structured

size is the size of the volume in 256-word blocks.

csr is the default value for the device's control and status register.

vec is the default value for the device's interrupt vector.

.DREND Macro Expansion

This macro generates the termination table for the termination section of a device handler.

Call: .DREND name

name is the 2-character device name.

The .DREND macro generates the following table:

```
$RLPTR:   .WORD 0 ($RELOC,XM only)
$MPPTR:   .WORD 0 ($MPPHY,XM only)
$GTBYT:   .WORD 0 ($GETBY,XM only)
$PTBYT:   .WORD 0 ($PUTBY,XM only)
$PTWRD:   .WORD 0 ($PUTWR,XM only)
$ELPTR:   .WORD 0 ($ERLOG,Error Logging)
$TIMIT:   .WORD 0 ($TIMIO,Device Timeout)
$INPTR:   .WORD 0 ($INTEN)
$FKPTR:   .WORD 0 ($FORK)
```

.DRFIN Macro Expansion

This macro generates the position-independent instructions for the jump back to the monitor at the end of a handler's I/O completion section. It also makes the pointer to the current queue element, ddCQE, a global symbol. When control passes to the monitor after the jump, the monitor releases the current queue element.

Call: .DRFIN name

name is the 2-character device name.

.DRSET Macro Expansion

This macro sets up, in block 0 of a device handler, the option table for a SET command. Use once for each SET option.

Call: .DRSET option, val, rtn [, mode]

option — is the name of the SET option. It can be up to six alphanumeric characters long with no embedded spaces or tabs.

val — is a parameter that is passed to the routine in R3. It must not be 0.

rtn — is the name of the routine that modifies the code in block 1 of the handler. The routine must follow the option table in block 0 and must not go above address 776. When the routine is entered, R0, R1, and R3 contain:

R0 = numeric value, if any, from SET command line
R1 = unit number; if none specified, bit 15 is set
R3 = the parameter *val*

mode — is an optional argument to indicate the type of SET parameter.

NO — a NO prefix is valid.
NUM — a decimal number is required.
OCT — an octal number is required.

The .DRSET macro generates the following table:

Offset	Contents
0	Value to pass in R3 to SET routine
2	Radix–50 of option name (first word)
4	Radix–50 option name (second word)
6	100 = decimal argument required 140 = octal argument required 200 = NO prefix is valid
7	Pointer to SET routine in handler

.DRVTB Macro Expansion

This macro sets up a table of 3-word entries for each vector of a multivector device. Use .DRVTB once for each vector. The table entries contain the vector location, interrupt entry point, and processor status word.

Call: .DRVTB name, vec, int [, ps]

name	is the 2-character device name. If the device has multiple vectors, leave this argument blank after the first use of .DRVTB in the handler.
vec	is the address of the vector, and must be from 0 to 474.
int	is the symbolic name of the interrupt handling routine. This is generally of the form *ddINT*, where *dd* is the 2-character device name.
ps	is an optional value that specifies the low-order four bits of the new processor status word in the interrupt vector. It defaults to 0 if omitted.

.DSTATUS EMT 342

This request obtains information about a particular device. If .DSTATUS is successful, it returns four words of status information starting at the address specified by addr.

Call: .DSTATUS addr, dnam

addr	is the address of a 4-word block that stores the status information.
dnam	is the address of a word containing the Radix–50 device name.

Errors:

Code	Explanation
0	Device was not found in tables.

.DSTATUS Information Block

Word 0:	Device	Characteristics
	(0–7)	Contains the RT–11 device code
VARSZ$	400	0 = .SPFUN 373 requests are invalid for this handler 1 = .SPFUN 373 requests (return volume size) are valid for this handler (VARSZ$ = 1 forces SPFUN$ = 1)
ABTIO$	1000	0 = Handler not entered at abort entry on normal program exits 1 = Handler entered at abort entry whenever a program terminates
SPFUN$	2000	0 = No .SPFUN requests accepted 1 = Handler accepts .SPFUN requests
HNDLR$	4000	0 = Handler abort entry taken only if an active queue element exists for aborted program 1 = Handler abort entry taken every time program is aborted
SPECL$	10000	1 = Non-RT–11 directory-structured device
WONLY$	20000	1 = Write-only device
RONLY$	40000	1 = Read-only device
FILST$	100000	0 = Sequential access device 1 = Random access device

Word 1: Handler Size

The size of the handler in bytes.

Word 2: Load address +6

If word 2 is 0, the handler is not in memory and must be fetched before it can be used. The address returned is the load address of the handler +6.

Word 3: Volume Size

The size in 256-word blocks of the volume for block-replaceable devices. This will be the smallest size volume for variable-size devices, or 0 for sequential devices. The last block on the volume is the volume size –1.

.ELAW EMT 375, Code 36 (XM Only)

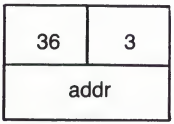
This request eliminates a virtual address window. An implied unmapping of the window occurs when its definition block is eliminated.

Call: .ELAW area [, addr]

area is the address of a 2-word EMT argument block.

addr is the address of the window definition block for the window to be eliminated. If the second word of the EMT argument block is already filled in, *addr* need not be specified.

Format: R0 –>



Errors:

Code	Explanation
3	An invalid window identifier was specified.

.ELRG EMT 375, Code 36 (XM Only)

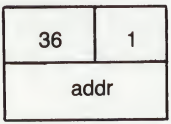
This request eliminates a dynamic region in physical memory and returns the memory to the free list where it can be used by other jobs.

Call: .ELRG area [, addr]

area is the address of a 2-word EMT argument block.

addr is the address of the region definition block for the region to be eliminated. Windows mapped to this region are unmapped. The static region, region 0, cannot be eliminated. If the second word of the EMT argument block is aready filled in, addr need not be specified.

Format: R0 –>



Errors:

Code	Explanation
2	An invalid region identifier was specified.

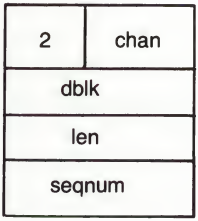
.ENTER EMT 375, Code 2

For file-structured devices, this request allocates space on the volume, creates a tentative entry in the volume's directory for the new file, and assigns the specified I/O channel number to the new file. For sequential devices such as a line printer, .ENTER assigns the I/O channel number to the device.

Call: .ENTER area, chan, dblk, len [, seqnum]

area	is the address of a 4-word EMT argument block.
chan	is an unused channel number in the range 0 to 376(octal).
dblk	is the address of a 4-word Radix–50 descriptor of the file to be created. If this word is 0, a non-file-structured .LOOKUP is done on the device.
len	is the file size specification.
seqnum	is a file sequence number for magtape or cassettes.

Format: R0 –>



Errors:

Code	Explanation
0	Channel is in use.
1	In a fixed-length request, no space greater than or equal to amount requested was found, or the volume's directory was full.
2	Non-sharable device is already in use by another job.
3	A file with that name already exists and is protected. A new file was not opened.
4	File sequence number was not found.
5	File sequence number is invalid or file name is null.

On return, R0 contains the size of the area allocated for use on a file-structured device. R0 returns 0 for sequential or non-file-structured devices.

.EXIT EMT 350

The .EXIT request terminates a user program. When used from a background job under the FB or XM monitors, or in SJ, .EXIT causes KMON to run in the background area. All outstanding mark time requests are canceled. Any I/O requests and/or completion routines pending for that job are allowed to complete. If R0 = 0, an implicit .HRESET is executed when KMON is entered.

Call: .EXIT

Errors:

None

The .EXIT request allows a user program to pass command lines to KMON in the chain information area (locations 500 to 777) for execution after the job exits. This is performed under the following conditions:

1) Location 510 contains the total number of bytes in the command lines to be passed to KMON.
2) Command lines are stored beginning at location 510 and must be .ASCIZ strings with no embedded carriage return or linefeed characters.
3) The user program sets bit 11 in the job status word immediately before an .EXIT issued with R0 = 0. In addition, if bit 5 of the JSW is set before the .EXIT, the status of any currently active indirect command file will be preserved.

.FETCH EMT 343

This request loads device handlers into memory from the system volume. This request is valid only for background jobs.

Call: .FETCH addr, dnam

addr is the address where the device handler is to be loaded.
dnam is the pointer to the Radix–50 device name.

Errors:

Code	Explanation
0	Either the device name specified is not installed in the system, or there is no handler for that device on the system volume.

On return, R0 points to the first available location above the handler.

.FORK Macro Expansion

Use this macro in a device handler or interrupt service routine when access to a shared resource must be serialized or when a lengthy but non-time-critical section of code must be executed.

Call: .FORK fkblk

fkblk is a 4-word block of memory allocated within the handler.

Errors:

None

The .FORK macro expands as follows:

```
.FORK   fkblk
JSR     R5,@$FKPTR
.WORD   fkblk-.
```

The .FORK macro issues a subroutine call to the monitor and does not use an EMT request. The .FORK call must be preceded by an .INTEN call, and the address of a 4-word block must be supplied with the request. This block is used as the fork queue element by the monitor. The program must not have left any information on the stack between the .INTEN and the .FORK call. The contents of registers R4 and R5 are preserved through the call, and on return registers R0 through R3 are available for use. The .FORK returns to the calling point with the interrupt dismissed and the priority at 0.

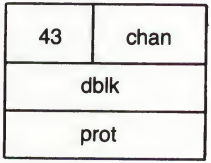
.FPROT EMT 375, Code 43

This request sets or removes file protection status for an individual file. Protected files cannot be deleted by .DELETE, .ENTER, or .RENAME programmed requests.

Call: .FPROT area, chan, dblk [, prot]

area	is the address of a 3-word EMT argument block.
chan	is a free channel number in the range 0 to 376(octal).
dblk	is the address of a 4-word block with the filespec in Radix–50 of the file whose protection is to be set or removed.
prot	value of 1 protects the file from deletion; value of 0 or omitted value removes the file's protection.

Format: R0 –>



Errors:

Code	Explanation
0	Channel is in use.
1	File was not found.
3	Operation is invalid.

.GMCX EMT 375, Code 36 (XM Only)

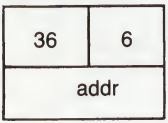
This request returns the mapping status of an extended memory window. Status is returned in the window definition block, and can be used in a subsequent mapping operation.

Call: .GMCX area [, addr]

area is the address of a 2-word EMT argument block.

addr is the address of the window definition block where the specified window's status is returned. If the second word of the EMT argument block is already filled in, *addr* need not be specified.

Format: R0 –>



Errors:

Code	Explanation
3	An invalid window identifier was specified.

.GTIM EMT 375, Code 21

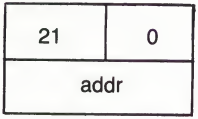
This request allows user programs to access the current time of day. The time is returned in two words as the number of clock ticks past midnight. The high-order time is returned in the first word, the low-order time in the second word.

Call: .GTIM area, addr

area is the address of a 2-word EMT argument block.

addr is the address of the 2-word area where the time is to be returned.

Format: R0 –>



Errors:

None

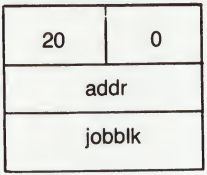
.GTJB EMT 375, Code 20

This request returns information about a job in the system.

Call: .GTJB area, addr [, jobblk]

area	is the address of a 3-word EMT argument block.
addr	is the address of an 8- or 12-word block into which the job parameters are passed.
jobblk	is a pointer to a 3-word ASCII job name for which data is being requested. If jobblk is omitted or is #-3, only the first eight words of information are returned.

Format: R0 –>



Errors:

Code	Explanation
0	No such job is currently running.

Information Returned:

Word	Offset	Contents
1	0	Job number (priority*2)
2	2	High-memory limit
3	4	Low-memory limit
4	6	Pointer to I/O channel space
5	10	Address of job's impure area
6	12	Low byte: unit number of job's console terminal High byte: reserved
7	14	Virtual high limit
8–9	16–20	Reserved
10–12	22–27	ASCII logical job name

.GTLIN EMT 345

This request collects a line of input from either the console terminal or an indirect command file, if one is active. It requires the USR, but the format of the input line is not checked.

In a foreground/background environment, calling .GTLIN performs a temporary and implicit .UNLOCK while the command line is being read.

Call: .GTLIN linbuf [, prompt] [,type]

linbuf	is the address of the buffer to receive the input line. The size of *linbuf* should be 81(decimal) bytes. The input line is stored in this area, terminated with a 0 byte.
prompt	is an optional argument and is the address of a prompt string to be printed on the console terminal. The prompt string must have the same format as the argument of a .PRINT request.
type	is an optional argument that forces .GTLIN to accept input only from the console terminal.

Errors:

Code	Explanation
0	Input line is longer than 80 characters.

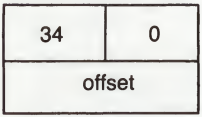
.GVAL EMT 375, Code 34

This request returns a monitor offset value in R0. This request must be used in an XM monitor environment to access the monitor's fixed offset locations, and should be used with all monitors to ensure compatibility. Refer to .PVAL.

Call: .GVAL area, offset

area	is the address of a 2-word EMT argument block.
offset	is the displacement from the beginning of the monitor to the word to be returned to R0.

Format: R0 –>



Errors:

Code	Explanation
0	The offset requested is beyond the limits of the resident monitor.

.HERR EMT 374, Code 5

This request turns off user interception of monitor errors. It allows the system to abort the job on fatal errors and generate an error message. This is the default case unless a .SERR has been done.

Call: .HERR

Format: R0 =

5	0

Errors:

None

.HRESET EMT 357

This request stops all I/O transfers in progress for the issuing job, and then performs an .SRESET request. The SJ monitor uses a hardware RESET instruction to terminate I/O.

Call: .HRESET

Errors:

None

.INTEN Macro Expansion

This request is used by device handler interrupt service routines to notify the monitor that an interrupt has occurred, to switch to system state, to set the processor priority to the correct value, and to save the contents of R4 and R5 before returning to the interrupt service routine.

Call: .INTEN prio [, pic]

prio	is the processor priority at which to run the interrupt routine.
pic	is an optional argument that should be specified if the interrupt routine is written as a PIC routine.

Errors:

None

The .INTEN request issues a subroutine call to the monitor and does not use an EMT request. All external interrupts must raise the processor to priority level 7. Use .INTEN to lower the priority to the value at which the device should be run. On return from .INTEN, the device interrupt can be serviced, at which point the interrupt routine can exit with an RTS PC.

.LOCK EMT 346

This request keeps the USR in memory to provide services required by the user program. In the FB monitor, a .CSIGEN, .CSISPC, or .GTLIN request performs an implicit and temporary .UNLOCK.

Call: .LOCK

Errors:

None

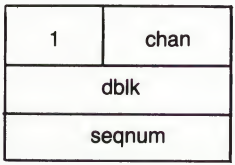
.LOOKUP EMT 375, Code 1

This request associates a specified channel with a device and file for I/O operations. The handler for the selected device must be in memory for this request.

Call: .LOOKUP area, chan, dblk [, seqnum]

area	is the address of a 3-word EMT argument block.
chan	is a channel number in the range 0 to 376(octal).
dblk	is the address of a 4-word Radix–50 descriptor of the device and file to be accessed.
seqnum	is a file number for magtapes and cassettes.

Format: R0 –>



Errors:

Code	Explanation
0	Channel is in use.
1	File indicated was not found on the volume.
2	File already open on a non-sharable device; for example, magtape.
5	Argument is invalid.

On return, R0 contains the length in blocks of the file just opened. On return from a .LOOKUP for a non-directory, file-structured device, R0 contains 0 for the length.

.MAP EMT 375, Code 36 (XM Only)

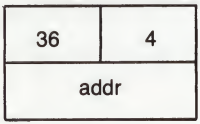
This request maps a previously defined address window into a dynamic region of extended memory or into the static region in the lower 28K. If the second word of the EMT argument block is aready filled in, *addr* need not be specified.

Call: .MAP area [, addr]

area is the address of a 2-word EMT argument block.

addr is the address of the window definition block containing a description of the window to be mapped and the region to which it will map.

Format: R0 –>



Errors:

Code	Explanation
2	An invalid region identifier was specified.
3	An invalid window identifier was specified.
4	The specified window was not mapped because the offset is beyond the end of the region, the region is larger than the window, or the window would extend beyond the bounds of the region.

.MFPS Macro Expansion

This macro call allows processor-independent user access to the processor status word. The .MFPS call is used to read the priority bits only. Condition codes are destroyed during the call and must be directly accessed prior to using the .MFPS macro.

Call: .MFPS [addr]

addr is the address into which the processor status is to be stored; if *addr* is not present, the value is returned on the stack. Only the priority bits are significant.

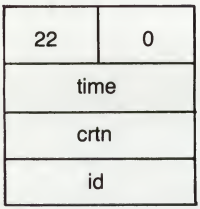
.MRKT EMT 375, Code 22

This request schedules a completion routine to be entered after a specified time interval (measured in clock ticks) has elapsed.

Call: .MRKT area, time, crtn, id

area	is the address of a 4-word EMT argument block.
time	is the address of a 2-word block containing the time interval (high order first, low order second) specified as a number of clock ticks.
crtn	is the entry point of a completion routine.
id	is a non-0 number assigned by the user to identify the particular request to the completion routine and to any .CMKT requests. The number must not be in the range 177000 to 177777, which is reserved for system use.

Format: R0 –>



Errors:

Code	Explanation
0	No queue element was available.

On entry to the completion routine, R0 contains the *id* number.

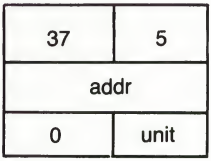
.MTATCH EMT 375, Code 37

This request attaches a terminal for exclusive use by the requesting job.

Call: .MTATCH area, addr, unit

area	is the address of a 3-word EMT argument block.
addr	is the optional address of an asynchronous terminal status (AST) word or it must be #0. (The AST word is a SYSGEN option.)
unit	is the logical unit number of the terminal.

Format: R0 –>



Errors:

Code	Explanation
2	Logical unit number does not exist.
3	Function code is out of range.
4	Unit is attached by another job; other job's number is returned in R0.
5	In the XM monitor, the status word address is not in valid user virtual address space.

Asynchronous Terminal Status Word (T.AST)

	(0–5)	Reserved
AS.HNG	100	Remote line hung up
AS.CAR	200	Carrier is present
	(8–12)	Reserved
AS.OUT	20000	Output ring buffer is empty
AS.INP	40000	Input is available from terminal
AS.CTC	100000	Multiple CTRL/C s typed at terminal

.MTDTCH EMT 375, Code 37

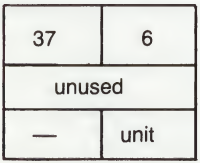
This request detaches a terminal from one job and makes the terminal available for other jobs.

Call: .MTDTCH area, unit

area is the address of a 3-word EMT argument block.

unit is the logical unit number (lun) of the terminal to be detached.

Format: R0 –>



Errors:

Code	Explanation
1	Unit number is invalid; unit is not attached.
2	Unit does not exist.
3	Request is invalid; function code is out of range.

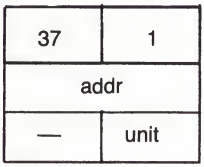
.MTGET EMT 375, Code 37

This request returns the status of the specified terminal unit to the caller.

Call: .MTGET area, addr, unit

area	is the address of a 3-word EMT argument block.
addr	is the address of a 4-word block where the status information is returned.
unit	is the logical unit number (lun) of the terminal for which status is requested. A unit need not be attached to the job issuing a .MTGET request.

Format: R0 –>



Errors:

Code	Explanation
1	Unit number is invalid; unit is not attached.
2	Unit does not exist.
3	Request is invalid; function code is out of range.
4	Unit is attached by another job.
5	With the XM monitor, the status block address is not in valid user virtual address space.

If an .MTGET request fails because the terminal is attached by another job, the job number of the owner is returned in R0 and the terminal status is returned. For all other error conditions, R0 is undefined. The .MTGET request returns the multiterminal status block.

Multiterminal Status Block

	M.TSTS		0
	M.TST2		2
5	M.FCNT	M.TFIL	4
7	M.TSTW	M.TWID	6

Terminal Configuration Word 1 (M.TSTS, offset = 0)

1	Hardware tab
2	Output carriage return/linefeed when carriage width exceeded
4	Hardware formfeed
10	CTRL/F, CTRL/B, and CTRL/X processed as normal characters
(4–5)	Reserved
100	Inhibit TT wait
200	XON/XOF processing enabled
7400	Line speed (baud rate) mask
10000	Character mode input
20000	Terminal is remote
40000	Lowercase to uppercase conversion disabled
100000	Backspace used for character deletion

Line Speed Mask (bits 8 to 11 in M.TSTS)

Mask	Speed	Mask	Speed
0000	50	4000	1800
0400	75	4400	2000
1000	110	5000	2400
1400	134.5	5400	3600
2000	150	6000	4800
2400	300	6400	7200
3000	600	7000	9600
3400	1200	7400	unused

Multiterminal Status Block (cont.)

Terminal Configuration Word 2 (M.TST2, offset = 2)

3	Character length (0–1)
4	Unit stop
10	Parity enable
20	Odd parity
(5–6)	Reserved
200	Read pass all
(8–14)	Reserved
100000	Write pass all

Terminal State Byte (M.TSTW, offset = 7)

1	Fill sequence in progress
2	CTRL/U in progress
(2–3)	Reserved
20	Detach in progress
40	.PRINT/.WRITE synch flag
100	Output interrupt expected
200	XON/XOFF processing enabled
(8–9)	Reserved
2000	Terminal is shared console
4000	Terminal has hung up
10000	Terminal interface is DZ11
(13)	Reserved
40000	Double CTRL/C was struck
100000	Terminal is acting console

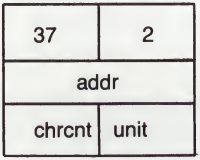
.MTIN EMT 375, Code 37

This request reads characters from the keyboard buffer. The .MTIN request is the multiterminal form of the .TTYIN request. The .MTIN request moves one or more characters to the buffer specified.

Call: .MTIN area, addr, unit [, chrcnt]

area	is the address of a 3-word EMT argument block.
addr	is the byte address of the user buffer.
unit	is the logical unit number of the terminal input.
chrcnt	is a character count indicating the number of characters to transfer. The valid range is from 0 to 255. A count of 0 transfers one character.

Format: R0 –>



Errors:

Code	Explanation
0	No input is available.
1	Unit number is invalid; unit is not attached.
2	Logical unit number does not exist.
3	Request is invalid; function code is out of range.
5	With the XM monitor, the user buffer address is not in valid user virtual address space.

On return, R0 contains the updated buffer address, pointing past the last character transferred.

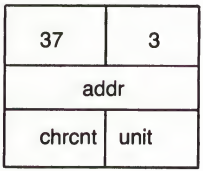
.MTOUT EMT 375, Code 37

This request transfers characters to the terminal output buffer. This request is the multiterminal form of the .TTYOUT request. The .MTOUT request moves one or more characters from the user's buffer to the output ring buffer of the terminal.

Call: .MTOUT area, addr, unit [, chrcnt]

area	is the address of a 3-word EMT argument block.
addr	is the address of the caller's input buffer.
unit	is the unit number of the terminal.
chrcnt	is a character count indicating the number of characters to transfer (1 to 255.).

Format: R0 –>



Errors:

Code	Explanation
0	Output buffer is full.
1	Unit number is invalid; unit is not attached.
2	Unit number does not exist.
3	Request is invalid; function code is out of range.
5	With the XM monitor, the user buffer address is not in valid user virtual address space.

On return, R0 contains the updated buffer address, pointing past the last character transferred.

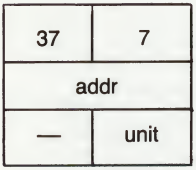
.MTPRNT EMT 375, Code 37

This request prints one or more lines on the specified terminal. It is the multiterminal form of the .PRINT request.

Call: .MTPRNT area, addr, unit

area	is the address of a 3-word EMT argument block.
addr	is the starting address of the character string to be printed. The string must be terminated with a null byte or a 200 byte.
unit	is the unit number associated with the terminal.

Format: R0 –>



Errors:

Code	Explanation
1	Unit number is invalid; unit is not attached.
2	Unit number does not exist.
5	With the XM monitor, the character string address is not in valid user virtual address space.

.MTPS Macro Expansion

This macro call allows processor-independent user access to the processor status word. The .MTPS request sets the priority bits only.

Call: .MTPS addr

addr is the address of the word to be placed in the processor status word. If *addr* is not present, the processor status word is taken from the stack. The high byte on the stack is set to 0 when *addr* is present.

.MTRCTO EMT 375, Code 37

This request resets the CTRL/O switch of the specified terminal. It is the multiterminal form of the .RCTRLO request.

Call: .MTRCTO area, unit

area is the address of a 3-word EMT argument block.

unit is the unit number associated with the terminal.

Format: R0 –>

37	4
unused	
—	unit

Errors:

Code	Explanation
1	Unit number is invalid; unit is not attached.
2	Unit does not exist.
3	Request is invalid; function code is out of range.

.MTSET EMT 375, Code 37

This request sets terminal and line characteristics for a terminal. It also determines the input/output mode of the terminal's service requests.

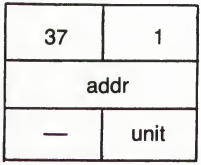
Call: .MTSET area, addr, unit

area is the address of a 3-word EMT argument block.

addr is the address of a 4-word status block containing the line and terminal status being set. Refer to the description following the .MTGET request.

unit is the logical unit number associated with the line and terminal.

Format: R0 –>



Errors:

Code	Explanation
1	Unit number is invalid; unit is not attached.
2	Unit does not exist.
3	Request is invalid; function code is out of range.
5	With the XM monitor, the status block address is not in valid user virtual address space.

.MTSTAT EMT 375, Code 37

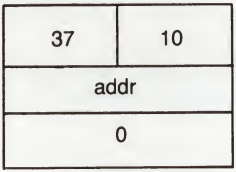
This request returns multiterminal system status information.

Call: .MTSTAT area, addr

area is the address of a 3-word EMT argument block.

addr is the address of an 8-word status block where multiterminal status information is returned.

Format: R0 –>



Errors:

Code	Explanation
5	With the XM monitor, the status block address is not in valid user virtual address space.

Information returned in the status block:

Offset	Contents
0	Offset from RMON to first TCB
2	Offset from RMON to console TCB
4	Number of TCBs in the system (1 to 17 octal)
6	Size of TCB in bytes
(10–17)	Reserved

.MWAIT EMT 374, Code 11 (FB and XM Only)

This request suspends execution of the job issuing the request until all messages sent to or requested from another job have been received.

Call: .MWAIT

Format: R0 =

11	0

Errors:

None

.PEEK EMT 375, Code 34

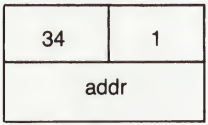
This request returns the contents of a location in low memory (0 to 28K). RT–11 utility programs use this request to examine locations in the monitor. Refer to .POKE.

Call: .PEEK area, addr

area is the address of a 2-word EMT argument block.

addr is the address of the location to examine. The contents of the location are returned in R0.

Format: R0 –>



Errors:

None

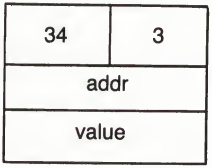
.POKE EMT 375, Code 34

This request deposits a value in a low memory location (0 to 28K). Refer to .PEEK.

Call: .POKE area, addr, value

area	is the address of a 3-word EMT argument block.
addr	is the address of the location to modify.
value	is the new value to deposit in the location. The old contents of the location are returned on R0.

Format: R0 –>



Errors:

None

.PRINT EMT 351

This request prints output on the console terminal. String printed can be terminated with either a 0 byte or an octal 200 byte. A carriage return/line feed combination is appended to a string ending in a 0 byte, while a string ending in an octal 200 byte leaves the cursor or carriage at the end of the printed line.

Call: .PRINT addr

addr is the address of the string to be printed.

Errors:

None

.PROTECT EMT 375, Code 31 (FB and XM Only)

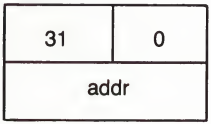
The .PROTECT request gives a job exclusive control of a 2-word vector in the region 0 to 476.

Call: .PROTECT area, addr

area is the address of a 2-word EMT argument block.

addr is the address of the word pair to be protected. The argument *addr* must be a multiple of 4, and must be less than or equal to 474(octal). The two words at *addr* and *addr*+2 are protected.

Format: R0 –>



Errors:

Code	Explanation
0	Protect failure; locations are already in use.
1	Address is greater than 474 or not a multiple of 4.

.PURGE EMT 374, Code 3

This request frees a channel without taking any other action.

Call: .PURGE chan

chan is the channel number to be purged.

Format: R0 =

3	chan

Errors:

None

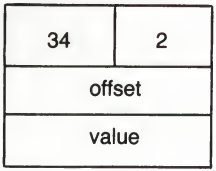
.PVAL EMT 375, Code 34

This request changes the contents of a monitor offset. Refer to .GVAL.

Call: .PVAL area, offset, value

area	is the address of a 3-word EMT argument block.
offset	is the displacement from the beginning of the monitor to the word to change.
value	is the new value to put in the location. The old contents of the location are returned in R0.

Format: R0 –>



Errors:

Code	Explanation
0	The offset requested is beyond the limits of the resident monitor.

This macro symbolically defines I/O queue element offsets. The .QELDF macro defines the following values:

Q.LINK	= 0	Link to next queue element
Q.CSW	= 2	Pointer to channel status word
Q.BLKN	= 4	Physical block number
Q.FUNC	= 6	Special function code
Q.JNUM	= 7	Job number
Q.UNIT	= 7	Device unit number
Q.BUFF	= 8.	User virtual memory buffer address
Q.WCNT	= 10.	Word count
Q.COMP	= 12.	Completion routine code

Since handlers usually deal with queue element offsets relative to Q.BLKN, the .QELDF macro also defines the following symbolic offsets:

Q$LINK	= –4
Q$CSW	= –2
Q$BLKN	= 0
Q$FUNC	= 2
Q$JNUM	= 3
Q$UNIT	= 3
Q$BUFF	= 4
Q$WCNT	= 6
Q$COMP	= 8.

For SJ and FB systems:

Q.ELGH	= 14.	Length of queue element

For XM systems:

Q.PAR	= 14.	PAR1 relocation bias
Q$PAR	= 10.	
Q.ELGH	= 20.	Length of queue element

The .DRDEF macro automatically calls .QELDF.

.QSET EMT 353

This request adds entries to the RT–11 I/O queue. The I/O queue initially has one queue element. Each program should allocate one more queue element than the total number of I/O requests that will be pending simultaneously on different channels.

Call: .QSET addr, len

addr is the address at which the new elements are to start.

len is the number of entries to be added. In the SJ and FB monitors, each queue entry is 7 words long. In the XM monitor, each queue entry is 10(decimal) words long.

Errors:

None

On return, R0 contains the address of the first word beyond the allocated queue elements.

.RCTRLO EMT 355

This request ensures that the console terminal is able to print by resetting the CTRL/O switch for the terminal. The .RCTRLO request should also be issued whenever any bits are changed in the JSW. This allows the monitor to update its internal status information to reflect the new contents of the JSW.

Call: .RCTRLO

Errors:

None

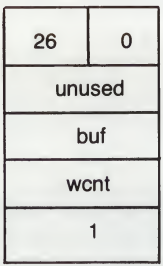
This request queues a request to receive a message or data sent by another job in FB or XM environments. Execution of the job issuing the request continues. Use .MWAIT to wait until the message has been received.

Note that .RCVDx and .SDATx use the same channel. A program should not issue synchronous (.RCVDW and .SDATW) requests and asynchronous (.RCVDC and .SDATC) requests at the same time because the synchronous requests will never return.

Call: .RCVD area, buf, wcnt

area	is the address of a 5-word EMT argument block.
buf	is the address of the buffer to which the message is to be sent.
wcnt	is the number of words to be transferred.

Format: R0 –>



Errors:

Code	Explanation
0	No other job exists in the system.

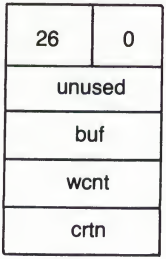
This request queues a request to receive a message or data sent by another job in an FB or XM environment and specifies a completion routine to be entered when the message is received. Execution of the job issuing the request continues.

Note that .RCVDx and .SDATx use the same channel. A program should not issue synchronous (.RCVDW and .SDATW) requests and asynchronous (.RCVDC and .SDATC) requests at the same time because the synchronous requests will never return.

Call: .RCVDC area, buf, wcnt, crtn

area	is the address of a 5-word EMT argument block.
buf	is the address of the buffer to which the message is to be sent.
wcnt	is the number of words to be transmitted.
crtn	is the address of a completion routine to be entered.

Format: R0 –>



Errors:

Code	Explanation
0	No other job exists in the system.

.RCVDW EMT 375, Code 26 (FB and XM Only)

This request queues a request to receive a message or data sent by another job in FB or XM environments. Execution of the job issuing the request is suspended until all outstanding messages have been sent or received.

Note that .RCVDx and .SDATx use the same channel. A program should not issue synchronous (.RCVDW and .SDATW) requests and asynchronous (.RCVDC and .SDATC) requests at the same time because the synchronous requests will never return.

Call: .RCVDW area, buf, wcnt

area	is the address of a 5-word EMT argument block.
buf	is the address of the buffer to which the message is to be sent.
wcnt	is the number of words to be transmitted.

Format: R0 –>

26	0
unused	
buf	
wcnt	
0	

Errors:

Code	Explanation
0	No other job exists in the system.

.RDBBK Macro Expansion (XM Only)

This macro defines symbols and reserves space for the region definition block. .RDBBK automatically calls .RDBDF.

Call: .RDBBK rgsiz

rgsiz is the size of the dynamic region needed, expressed in 32-word units.

.RDBDF Macro Expansion (XM Only)

This macro defines the symbolic offset names for the region definition block and the names for the region status word bit patterns. .RDBDF also defines R.GLGH, the length of the region definition block.

Call:
.RDBDF

Expansion:

```
R.GID  =      0
R.GSIZ =      2
R.GSTS =      4
R.GLGH =      6
RS.NAL = 20000
RS.UNM = 40000
RS.CRR = 100000
```

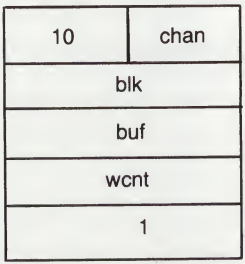
.READ EMT 375, Code 10

This request transfers to memory a specified number of words from the device associated with the specified channel. Control returns to the user program immediately after the .READ is initiated.

Call: .READ area, chan, buf, wcnt, blk

area	is the address of a 5-word EMT argument block.
chan	is a channel number in the range 0 to 376(octal).
buf	is the address of the buffer to receive the data read.
wcnt	is the number of words to read.
blk	is the starting block number to read.

Format: R0 –>



Errors:

Code	Explanation
0	Attempt was made to read past end of file.
1	Hard error occurred on channel.
2	Channel is not open.

On return, R0 contains the number of words transferred. This will be less than the number requested if end-of-file is reached on the input device.

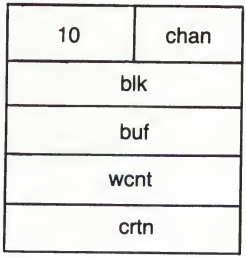
.READC EMT 375, Code 10

This request transfers a specified number of words from the channel to memory. Control returns to the user program immediately after the .READC is initiated. Control passes to the completion routine when the .READC is complete.

Call: .READC area, chan, buf, wcnt, crtn, blk

area	is the address of a 5-word EMT argument block.
chan	is a channel number in the range 0 to 376(octal).
buf	is the address of the buffer to receive the data read.
wcnt	is the number of words to read.
crtn	is the address of the completion routine and must be above 500(octal).
blk	is the block number to read.

Format: R0 –>



Errors:

Code	Explanation
0	Attempt was made to read past end-of-file.
1	Hard error occurred on channel.
2	Channel is not open.

On return, R0 contains the number of words transferred. This will be less than the number requested if end-of-file is reached on the input device.

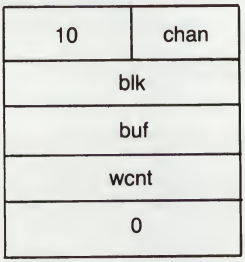
.READW EMT 375, Code 10

This request transfers a specified number of words from a channel to memory. When the .READW is complete or an error is detected, control returns to the user program.

Call: .READW area, chan, buf, wcnt, blk

area	is the address of a 5-word EMT argument block.
chan	is the channel number in the range 0 to 376(octal).
buf	is the address of the buffer to receive the data read.
wcnt	is the number of words to read.
blk	is the block number to be read.

Format: R0 –>



Errors:

Code	Explanation
0	Attempt was made to read past end-of-file.
1	Hard error occurred on channel.
2	Channel is not open.

On return, R0 contains the number of words transferred. This will be less than the number requested if end-of-file is reached on the input device.

.RELEAS EMT 343

This request notifies the monitor that a fetched device handler is no longer needed. The .RELEASE request is ignored if the handler is the system device handler, is not currently resident, or is resident because of a LOAD command to KMON.

Call: .RELEAS dnam

dnam is the address of the Radix–50 device name.

Errors:

Code	Explanation
0	Device name is invalid.

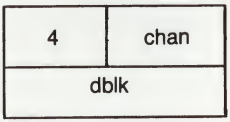
.RENAME EMT 375, Code 4

This request changes the name of a file and puts the current system date in the file's directory entry.

Call: .RENAME area, chan, dblk

area	is the address of a 2-word EMT argument block.
chan	is a channel number in the range 0 to 376(octal).
dblk	is the address of a block that specifies the file to be renamed followed by the new file name, in Radix–50.

Format: R0 –>



Errors:

Code	Explanation
0	Channel is open.
1	File was not found.
2	Operation is invalid (device is not file structured).
3	A protected file of the specified name already exists. The .RENAME is not performed.

.REOPEN EMT 375, Code 6

This request reassigns a channel to a .SAVESTATUS block. This allows I/O to continue on the channel from the point at which the .SAVESTATUS was performed.

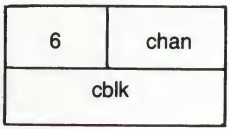
Call: .REOPEN area, chan, cblk

area is the address of a 2-word EMT argument block.

chan is a channel number in the range 0 to 376(octal).

cblk is the address of the 5-word block where the .SAVESTATUS information was stored.

Format: R0 –>



Errors:

Code	Explanation
0	The specified channel is in use. The .REOPEN is not performed.

.RSUM EMT 374, Code 2 (FB & XM Only)

This request resumes execution of a job's mainline code after a .SPND.

Call: .RSUM

Format: R0 =

Errors:

None

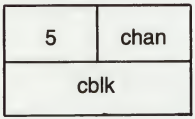
.SAVESTATUS EMT 375, Code 5

This request stores five words of channel status information in memory. The channel is freed upon completion of the request. The channel can be reopened, possibly with a different channel number, by a .REOPEN request.

Call: .SAVESTATUS area, chan, cblk

area	is the address of a 2-word EMT argument block.
chan	is a channel number in the range 0 to 376(octal).
cblk	is the address of the 5-word user memory block where the channel status information is to be stored.

Format: R0 –>



Errors:

Code	Explanation
0	The channel specified is not associated with any file.
1	The channel was opened with an .ENTER request, or a .SAVESTATUS request was attempted for a magtape or cassette file.

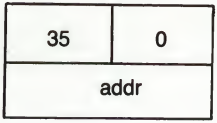
.SCCA EMT 375, Code 35

This request inhibits CTRL/C abort, indicates when a double CTRL/C is initiated at the console, and distinguishes between single and double CTRL/C commands. Bit 15 of the terminal status word is set when consecutive CTRL/C characters are detected. The user program must clear the bit.

Call: .SCCA area, addr

area is the address of a 2-word parameter block.
addr is the address of a terminal status word; an address of 0 re-enables the CTRL/C command.

Format: R0 –>



Errors:

None

.SDAT EMT 375, Code 25 (FB and XM Only)

This request queues a request to send a message or data to another job in FB or XM environments. Execution of the job issuing the request continues.

Note that .RCVDx and .SDATx use the same channel. A program should not issue synchronous (.RCVDW and .SDATW) requests and asynchronous (.RCVDC and .SDATC) requests at the same time because the synchronous requests will never return.

Call: .SDAT area, buf, wcnt

area	is the address of a 5-word EMT argument block.
buf	is the buffer address of the beginning of the message to transfer.
wcnt	is the number of words to transfer.

Format: R0 –>

25	0
unused	
buf	
wcnt	
1	

Errors:

Code	Explanation
0	No other job exists.

.SDATC EMT 375, Code 25 (FM and XM Only)

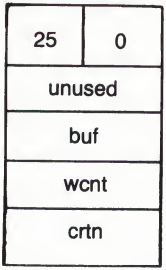
This request queues a request to send a message or data to another job in FB or XM environments and specifies a completion routine to be entered when the message has been transmitted. Execution of the job issuing the request continues.

Note that .RCVDx and .SDATx use the same channel. A program should not issue synchronous (.RCVDW and .SDATW) requests and asynchronous (.RCVDC and .SDATC) requests at the same time because the synchronous requests will never return.

Call: .SDATC area, buf, wcnt, crtn

area	is the address of a 5-word EMT argument block.
buf	is the buffer address of the beginning of the message to transfer.
wcnt	is the number of words to transfer.
crtn	is the address of the completion routine to enter when the message has been transmitted.

Format: R0 –>



Errors:

Code	Explanation
0	No other job exists.

.SDATW EMT 375, Code 25 (FB and XM Only)

This request queues a request to send a message or data to another job in FB or XM environments. Execution of the job issuing the request is suspended until all outstanding messages are sent and received.

Note that .RCVDx and .SDATx use the same channel. A program should not issue synchronous (.RCVDW and .SDATW) requests and asynchronous (.RCVDC and .SDATC) requests at the same time because the synchronous requests will never return.

Call: .SDATW area, buf, wcnt

area	is the address of a 5-word EMT argument block.
buf	is the buffer address of the beginning of the message to transfer.
wcnt	is the number of words to transfer.

Format: R0 –>

25	0
unused	
buf	
wcnt	
1	

Errors:

Code	Explanation
0	No other job exists.

.SDTTM EMT 375, Code 40

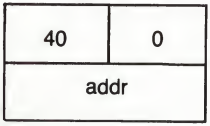
This request allows programs to set the system date and time.

Call: .SDTTM area, addr

area is the address of a 2-word EMT argument block.

addr is the address of a 3-word block containing the new date and time.

Format: R0 –>



Errors:

None

Date/Time information block:

Offset	Contents
0	Date in RT–11 format (–1 to leave date unchanged)
2	High-order time (–1 to leave time unchanged)
4	Low-order time

The double-precision time is expressed in ticks past midnight.

.SERR EMT 374, Code 4

This request inhibits job abort on monitor errors. If a monitor error occurs during a monitor request, the carry bit is set and byte 52 contains a negative value indicating the error condition that occurred.

Call: .SERR

Format: R0 =

Errors:

Code	Explanation
–1	USR was called from a completion routine.
–2	Operation attempted to access an unloaded device handler.
–3	I/O error occurred while accessing the directory.
–4	Error occurred during a .FETCH request.
–5	Error occurred while reading an overlay.
–6	Attempt was made to do a .ENTER on a full directory.
–7	Attempt was made to access an invalid address.
–10	Invalid channel number was specified.
–11	Invalid EMT request was issued.
–12	Reserved for use by monitor.
–13	Reserved for use by monitor.
–14	Directory on a device is invalid.
–15	Attempt was made to fetch an unloaded handler while running an XM monitor that does not have .FETCH support.
–16	Reserved for use by monitor.
–17	Reserved for use by monitor.
–20	Reserved for use by monitor.
–21	Reserved for use by monitor.
–22	Reserved for use by monitor.

.SETTOP EMT 354

This request specifies a new address as a program's upper limit.

Call: .SETTOP addr

addr is the address of the highest word of the area desired; that is, the last word the program will modify, not the first word it leaves untouched.

Errors:

None

On return, R0 and location 50 contain the highest memory address allocated for use.

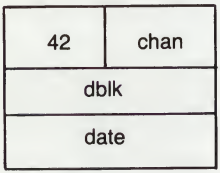
.SFDAT EMT 375, Code 42

This request allows user and utility programs to set or modify the creation date in a file's directory entry.

Call: .SFDAT area, chan, dblk [, date]

area	is the address of a 4-word EMT argument block.
chan	is an unused channel number in the range 0 to 376(octal).
dblk	is a 4-word block containing the filespec in Radix–50 of the file for which the date is to be modified.
date	is the address of the new date word in RT date format. If *date* is #0 or omitted, .SFDAT uses the current date. No check is made for an invalid date.

Format: R0 –>



Errors:

Code	Explanation
0	Channel is in use.
1	File was not found.
2	Operation is invalid (device is not file structured).
3	File is protected.

.SFPA EMT 375, Code 30

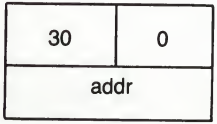
This request allows users with floating-point hardware to set trap addresses to be called when a floating-point exception occurs.

Call: .SFPA area, addr

area is the address of a 2-word EMT argument block.

addr is the address of the routine to enter when an exception occurs.

Format: R0 –>



Errors:

None

.SPCPS EMT 375, Code 41 (FB and XM Only)

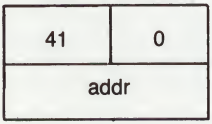
This request allows a program's completion routine to change the flow of control of the mainline code. It saves the mainline code PC and PS and changes the mainline PC to a new value. If the mainline code is performing a monitor request, the monitor allows the request to finish before doing any rerouting. The actual rerouting is deferred until the mainline code is about to run.

Call: .SPCPS area, addr

area is the address of a 2-word EMT argument block.

addr is the address of a 3-word block in user memory that contains the new mainline PC, and that is to contain the old mainline PC and PS.

Format: R0 –>



Errors:

Code	Explanation
0	The program issued the .SPCPS call from the mainline code rather than from a completion routine.
1	A previous .SPCPS request is outstanding.

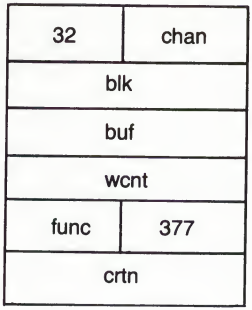
.SPFUN EMT 375, Code 32

This request is used with certain device handlers for device-dependent functions, such as rewind, backspace, read/write absolute sectors, and size volume.

Call: .SPFUN area, chan, func, buf, wcnt, blk [, crtn]

area	is the address of a 6-word EMT argument block.
chan	is a channel number in the range 0 to 376(octal).
func	is the numeric code of the function to be performed.
buf	is the buffer address; 0 if not used.
wcnt	is defined in terms of the device handler.
blk	is also defined in terms of the device handler.
crtn	is the entry point of a completion routine.

Format: R0 –>



Errors:

Code	Explanation
0	Attempt was made to read or write past end-of-file.
1	Hard error occurred on channel.
2	Channel is not open.

Function Codes for .SPFUN

MM,MS,MT	CT	
	377	Forward to last file
	376	Forward to last block
	375	Forward to next file
	374	Forward to next block
373	373	Rewind to load point
	372	Write file gap
377		Write EOF
376		Forward one block
375		Backspace one block
374		Write with extended gap
372		Off-line rewind
371		Write
370		Read
367		Stream at 100 ips (MS only)

DL,DM,DX,DY,LD

377	Read (DL,DM,DX,DY)
376	Write (DL,DM,DX,DY)
375	Write with deleted data mark (DX,DY)
374	Force read of bad block replacement table (DL,DM)
373	Return volume size (DL,DM,DY,LD)
372	Read/write handler translation table (LD)

For RK06/07 handler (DM), special function codes 377 and 376 require the buffer size to be one word larger than necessary for the data. The first word of the buffer contains the error information returned as a result of the .SPFUN request. Error codes and meanings are as follows:

100000	The I/O operation is successful.
100001	An ECC error is corrected.
100002	An error recovered on retry.
100004	An error recovered through an offset retry.
100010	An error recovered after recalibration.
100200	A bad block was detected (BSE error).
1774xx	An error did not recover.

.SPND EMT 374, Code 1 (FB and XM Only)

This request suspends the mainline program and allows only completion routines for I/O and mark time requests to run.

Call: .SPND

Errors:

None

.SRESET EMT 352

This request, software reset, performs the following functions:

Cancels any messages sent by the job
Waits for all I/O to complete
Dismisses any device handlers FETCHed
Purges any currently open files
Reverts to 16(decimal) I/O channels
Clears the job's .SPND/.RSUMM counter
Resets the I/O queue to one element
Cancels all outstanding .MRKT requests

Call: .SRESET

Errors:

None

.SYNCH Macro Expansion

This macro enables a program to issue programmed requests from within an interrupt service routine. Code following the .SYNCH call runs at priority level 0 as a completion routine in the issuing job's context.

Call: .SYNCH area [,pic]

area is the address of a 7-word block for use by .SYNCH.

pic is an optional argument which, if not blank, causes the .SYNCH macro to produce position-independent code.

Errors:

None

SYNCH Block Format:

Offset	Contents
0	Maintained by the monitor, do not alter
2	Current job number; must be set by program prior to issuing the .SYNCH
4	Reserved
6	Reserved
10	Synch ID
12	–1
14	0

The .SYNCH macro issues a subroutine call to the monitor and does not use an EMT request. Nothing may be placed on the stack between the .INTEN and .SYNCH calls. Normal return is to the word after the error return. R0 contains the Synch ID that was in word 5 of the block. R0 and R1 are free for use.

.TIMIO Macro Expansion

This macro issues a device time-out call in a device handler I/O initiation section. This request schedules a completion routine to run after the specified time interval has elapsed. The completion routine runs in the context of the job indicated in the timer block.

Call: .TIMIO tblk, hi, lo

tblk	is the address of a 7-word timer block.
hi	is the high-order time interval.
lo	is the low-order time interval.

Errors:

None

Timer block format:

Offset	Contents
0	High-order time word
2	Low-order time word
4	Link to next queue element; 0 indicates none
6	Owner's job number; 0 for background job, MAXJOB for foreground job, and (job priority)*2 for system jobs
10	Sequence number of timer request; valid range of sequence numbers is 177000 to 177377
12	–1
14	Address of the completion routine to execute if time-out occurs: the monitor zeroes this word when the completion routine is called, indicating that the timer block is available for reuse. When the completion routine is entered, R0 contains the sequence number of the request that timed-out.

.TLOCK EMT 374, Code 7

This request is used in an FB environment to gain ownership of the USR. The .TLOCK request is similar to the .LOCK request in that if successful, the user job returns with the USR in memory (identical to .LOCK in the SJ monitor). With .TLOCK, if the USR is not available, control returns immediately with the C-bit set to indicate the .TLOCK request failed.

Call: .TLOCK

Format: R0 =

Errors:

Code	Explanation
0	USR is already in use by another job.

.TRPSET EMT 375, Code 3

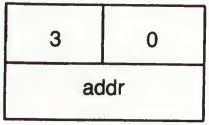
This request allows the user job to intercept traps to 4 and 10 instead of having the job aborted with a *?MON–F–Trap to nn* message. The carry bit determines which trap occurred: carry bit clear indicates a trap to 4; carry bit set indicates a trap to 10.

Call: .TRPSET area, addr

area is the address of a 2-word EMT argument block.

addr is the address of the trap routine. If an address of 0 is specified, trap interception is disabled.

Format: R0 –>



Errors:

None

.TTYIN/.TTINR EMT 340

These requests transfer a character from the console terminal to the user program. The character is passed as a right-justified byte in R0. The operation of .TTYIN/.TTINR can be altered by the setting of certain bits in the JSW. The expansion of .TTYIN is:

```
EMT     340
BCS     .–2
```

The expansion of .TTINR is:

```
EMT     340
```

If no characters are available when an EMT 340 is executed, return is made with the carry bit set.

Call: .TTYIN char

Call: .TTINR char

char is a pointer to a location where the character in R0 is to be stored. If char is specified, the character is returned both in R0 and the address pointed to by char. If char is not specified, the character is returned only in R0.

Errors:

Code	Explanation
0	No characters are available in the ring buffer.

.TTYOUT/.TTOUTR EMT 341

These requests transfer a character to the console terminal. If the monitor's output ring buffer has no room for the character, the .TTYOUT request waits for room before returning, but the .TTOUTR request returns immediately with carry set. The expansion of .TTYOUT is:

```
.EMT    341
BCS     .–2
```

The expansion of .TTOUTR is:

```
.EMT 341
```

Call: .TTYOUT char

Call: .TTOUTR char

char is the location containing the character to be loaded into R0 and printed. If not specified, the character in R0 is printed. Upon return from the request, R0 still contains the character.

Errors:

Code	Explanation
0	Output ring buffer is full.

.TWAIT EMT 375, Code 25

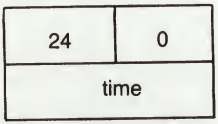
This request suspends the user's job for an indicated length of time. This request requires a queue element.

Call: .TWAIT area, time

area is the address of a 2-word EMT argument block.

time is a pointer to 2 words of time (high-order first, low-order second), expressed in ticks past midnight.

Format: R0 –>



Errors:

Code	Explanation
0	No queue element was available.

.UNLOCK EMT 347

This request releases the USR from memory if it was placed there with a .LOCK request. If the .LOCK required a swap, the .UNLOCK loads the user program back into memory.

Call: .UNLOCK

Errors:

None

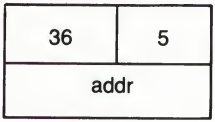
.UNMAP EMT 375, Code 36 (XM Only)

This request unmaps a window and flags that portion of the program's virtual address space as inaccessible. When an unmap operation is performed for a virtual job, attempts to access the unmapped address space cause a memory management fault.

Call: .UNMAP area, addr

area is the address of a 2-word argument block.
addr is the address of the window control block that describes the window to be unmapped.

Format: R0 –>



Errors:

Code	Explanation
3	An invalid window identifier was specified.
5	The specified window was not already mapped.

.UNPROTECT EMT 375, Code 31

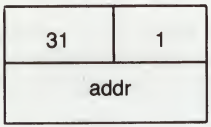
This request is the complement of the .PROTECT request. It cancels any protected vectors in the 0 to 476 area. An attempt to unprotect a vector that a job has not protected is ignored.

Call: .UNPROTECT area, addr

area is the address of a 2-word EMT argument block

addr is the address of the word pair to be canceled. The argument *addr* must be a multiple of 4 and must be less than or equal to 474(octal).

Format: R0 –>



Errors:

Code	Explanation
1	Address is greater than 474 or not a multiple of 4.

.WAIT EMT 374, Code 0

This request suspends program execution until all I/O requests on the channel are completed. The .WAIT request is typically combined with the .READ and .WRITE requests to perform double buffering.

Call: .WAIT chan

Format: R0 =

Errors:

Code	Explanation
0	Channel specified is not open.
1	Hardware error occurred on the previous I/O operation on this channel.

.WDBBK Macro Expansion (XM Only)

This macro defines symbols and reserves space for the window definition block. Information provided to the arguments of this macro permits the creation and mapping of a window through the use of the .CRAW request. .WDBBK automatically calls .WDBDF.

Call: .WDBBK wnapr, wnsiz [, wnrid, wnoff, wnlen, wnsts]

wnapr is the number of the Active Page Register (APR) set that includes the window's base address. A window must start on a 4K-word boundary. The valid range of values is 0 through 7.

wnsiz is the size of this window, expressed in 32-word units.

wnrid is the identification for the region to which this window maps.

wnoff is the offset into the region at which to start mapping this window, expressed in 32-word units.

wnlen is the amount of this window to map, expressed in 32-word units. The default value is 0, which maps as much of the window as possible.

wnsts is the window status word.

.WDBDF Macro Expansion (XM Only)

This request defines the symbolic offset names for the window definition block and the names for the window status word bit patterns. This macro also defines W.NLGH, the length of the window definition block.

Call: .WDBDF

Expansion:

```
W.NID   =      0
W.NAPR  =      1
W.NBAS  =      2
W.NSIZ  =      4
W.NRID  =      6
W.NOFF  =     10
W.NLEN  =     12
W.NSTS  =     14
W.NLGH  =     16
WS.MAP  =    400
WS.ELW  =  20000
WS.UNM  =  40000
WS.CRW  = 100000
```

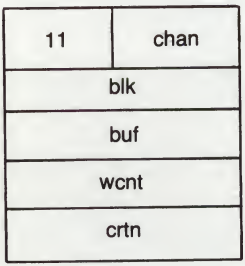
.WRITC EMT 375, Code 11

This request transfers a specified number of words from memory to a channel. Control returns to user program immediately after the request is queued. When the .WRITC completes, control passes to the completion routine specified in the request.

Call: .WRITC area, chan, buf, wcnt, crtn, blk

area	is the address of a 5-word EMT argument block.
chan	is a channel number in the range 0 to 376(octal).
buf	is the address of the memory buffer to be used for output.
wcnt	is the number of words to write.
crtn	is the address of the completion routine and must be above 500(octal).
blk	is the starting block number to write.

Format: R0 –>



Errors:

Code	Explanation
0	Attempt was made to write past end-of-file.
1	Hardware error occurred.
2	Channel is not open.

On return, R0 contains the number of words transferred. This will be less than the number requested if the output device has insufficient space.

.WRITE EMT 375, Code 11

This request transfers a specified number of words from memory to a channel. Control returns to user program immediately after the request is queued.

Call: .WRITE area, chan, buf, wcnt, blk

area	is the address of a 5-word EMT argument block.
chan	is a channel number in the range 0 to 376(octal).
buf	is the address of the memory buffer to be used for output.
wcnt	is the number of words to write.
blk	is the starting block number to write.

Format: R0 –>

11	chan
blk	
buf	
wcnt	
1	

Errors:

Code	Explanation
0	Attempt was made to write past end-of-file.
1	Hardware error occurred.
2	Channel is not open.

On return, R0 contains the number of words transferred. This will be less than the number requested if the output device has insufficient space.

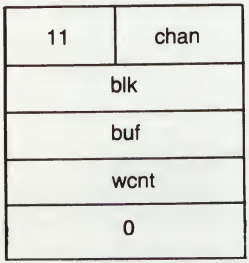
.WRITW EMT 375, Code 11

This request transfers a specified number of words from memory to a channel. Execution of the job is suspended until the .WRITW completes.

Call: .WRITW area, chan, buf, wcnt, blk

area	is the address of a 5-word EMT argument block.
chan	is a channel number in the range 0 to 376(octal).
buf	is the address of the memory buffer to be used for output.
wcnt	is the number of words to write.
blk	is the starting block number to write.

Format: R0 –>



Errors:

Code	Explanation
0	Attempt was made to write past end-of-file.
1	Hardware error occurred.
2	Channel is not open.

On return, R0 contains the number of words transferred. This will be less than the number requested if the output device has insufficient space.

Monitor Data Structures

Control Blocks

I/O Channel Block Format

Name	Offset	Contents
C.CSW	0	Channel status word
C.SBLK	2	Starting block number of file (0 if not file structured)
C.LENG	4	Length of file, if opened by .LOOKUP; size of empty area, if opened by .ENTER
C.USED	6	Highest block written
C.DEVQ	10	Number of pending requests (byte)
C.UNIT	11	Device unit number (byte)

Channel Status Word (C.CSW) Bit Definitions

Name	Bits	Meaning
HDERR$	1	Hard error
INDX$M	76	Index mask into $PNAME and other device tables
RENAM$	100	Rename operation is in progress
DWRIT$	200	File was opened with a .ENTER; monitor will modify directory when file is closed
DBLK$M	17400	Mask for directory segment containing this entry
EOF$	20000	End-of-file found on this channel
	(14)	Reserved
ACTIV$	100000	Channel is active

Handler Prefix Area Format

The handler prefix area is contained in block 0 of each handler.

Name	Offset	Contents
H.SIZ	52	Size of handler in bytes
H.DVSZ	54	Size of device in 256-word blocks
H.DSTS	56	Default device status value
H.GEN	60	Sysgen options
H.BPTR	62	Pointer to primary bootstrap
H.BLEN	64	Length of primary bootstrap in bytes
H.READ	66	Pointer to read routine for bootstrap

Region Control Block Format

Name	Offset	Contents
R.BADD	0	Starting physical address in 32-word blocks; 0 if not in use
R.BSIZ	2	Size of region in 32-word blocks
R.BSTA	4	Region status (byte)
R.BNWD	5	Number of windows mapped to region (byte)

Region Status Byte (R.BSTA) Bit Definitions

Name	Bits	Meaning
R.STOP	1	Region created by .SETTOP

Region Descriptor Block Format

Name	Offset	Contents
R.GID	0	Region ID returned by system
R.GSIZ	2	Size of region in 32-word blocks
R.GSTS	4	Region status word

Region Status Word (R.GSTS) Bit Definitions

Name	Bits	Meaning
	(0–12)	Reserved
RS.NAL	20000	Region was not previously allocated
RS.UNM	40000	One or more windows were unmapped
RS.CRR	100000	Region was successfully created

Window Control Block Format

Name	Offset	Contents
W.BRCB	0	Pointer to region control block of region to which window is mapped
W.BLVR	2	Low virtual address limit
W.BHVR	4	High virtual address limit
W.BSIZ	6	Size of window in 32-word blocks; 0 if not in use
W.BOFF	10	Offset into region in 32-word blocks
W.BFPD	12	First PDR in window (byte)
W.BNPD	13	Number of PDR's in window (byte)
W.BLPD	14	Contents of last PDR

Window Definition Block Format

Name	Offset	Contents
W.NID	0	Window identifier (byte)
W.NAPR	1	Base PAR for window (byte)
W.NBAS	2	Virtual base address
W.NSIZ	4	Window size in 32-word blocks
W.NRID	6	Region identifier
W.NOFF	10	Offset into region in 32-word blocks
W.NLEN	12	Length of window to map
W.NSTS	14	Window status word

Window Status Word (W.NSTS) Bit Definitions:

Name	Bits	Meaning
	(0–7)	Reserved
WS.MAP	400	Map window after creating it
	(9–12)	Reserved
WS.ELW	20000	One or more windows were eliminated
WS.UNM	40000	One or more windows were unmapped
WS.CRW	100000	Window successfully created

System Communications Area (SYSCOM)

Name	Address	Contents
USERPC	40	Start address of job
USERSP	42	Initial value of the stack pointer; default is 1000
JSW	44	Job Status Word
UFLOAT	46	USR load address
USERTO	50	High memory address of the user program
ERRBYT	52	EMT error reporting byte
USERRB	53	User program error reporting byte
SYSPTR	54	Address of the beginning of the Resident Monitor
TTFILL	56	Fill character
TTNFIL	57	Fill count

JSW Bit Definitions

Name	Bits	Meaning
	(0–2)	Reserved
GTLIN$	10	Non-terminating .GTLIN bit
EDIT$	20	If set, turns off single-line editor
SPXIT$	40	Special chain exit
TCBIT$	100	FB, XM; Inhibit terminal wait bit
HLTER	200	SJ; I/O Error halt
CHAIN$	400	CHAIN bit
OVLY$	1000	Overlaid program
VIRT$	2000	XM; Virtual image bit
CHNIF$	4000	Pass line to KMON bit
TTSPC$	10000	Special mode terminal bit
RSTRT$	20000	Reenter bit for programs
TTLC$	40000	Lowercase bit for console
USWAP$	100000	SJ; USR swap bit

USERRB Bit Definitions

Name	Bits	Meaning
SUCCS$	1	No error
WARN$	2	Warning
ERROR$	4	Error
SEVER$	10	Severe error
FATAL$	20	Fatal error

Block-Replaceable Device Directory Structure

Header:

Name	Offset	Contents
D.TOTA	0	Total number of segments in this directory; the valid range is 1 to 31(decimal)
D.NEXT	2	The segment number of the next logical directory segment; 0 if none
D.HIGH	4	The number of the highest segment currently in use; only maintained in first directory segment
D.EXTR	6	The number of extra bytes per directory entry; always an unsigned, even octal number
D.STRT	10	The block number on the device where the actual stored data monitored by this segment begins

Directory Entry:

Name	Offset	Contents
	0	Entry status word
E.NAME	2	File name; three words of Radix–50
E.LENG	10	Size in blocks of file, or size of free area if unused or tentative
E.USED	12	Unused on disk for permanent files; in memory, highest block written
E.CHAN	12	Channel number of tentative entry (byte)
E.JNUM	13	FB, XM; Job number of tentative entry (byte)
E.DATE	14	Creation date
	16+	Extra words, if any

Entry Status Word (Bit Definitions):

Name	Bits	Meaning
	(0–7)	Reserved
TENT	400	Tentative entry
EMPTY	1000	Empty entry
PERM	2000	Permanent entry
ENDBLK	4000	End of segment
	(12–14)	Reserved
PROT	100000	Protected entry (only if permanent)

Home Block Contents

Offset	Contents
000–201	Bad block replacement table
204–251	INITIALIZE/RESTORE data area
252–273	BUP information area
700–701	Reserved
702–703	Block number of first user file; always 0
722–723	Pack cluster size; always 1
724–725	Block number of first directory segment; always 6
726–727	System version in Radix–50
730–743	Volume identification
744–757	Owner name
760–773	System identification
776–777	Checksum

Bad Block Replacement Table Entries

Offset	Contents
0	Bad block number; 0 if end of table
2	Replacement block number

SAV Image File Format (Block 0)

Offset	Contents
0	.RAD50 /VIR/ for virtual overlaid files
2	Virtual high limit for virtual overlaid files
4	Reserved
6	Reserved
10	Reserved
12	Reserved
14	XM only; BPT trap
16	XM only; BPT trap
20	XM only; IOT trap
22	XM only; IOT trap
24	Reserved
26	Reserved
30	Reserved
32	Reserved
34	Trap vector (TRAP)
36	Trap vector (TRAP)
40	Relative start address
42	Initial stack pointer
44	Job Status Word (JSW)
46	USR swap address
50	Program's high limit
52	Size of root segment; used for .REL files only
54	Stack size in bytes; used for .REL files only
56	Size of overlay region; used for .REL files only
60	.RAD50 /REL/; used for .REL files only
62	Relative block of start of relocation information; used for .REL files only
64	Address of overlay handler table for overlaid files
66	Address of start of window definition blocks, for virtual overlaid files
70–356	Reserved
360–377	Memory usage bitmap

RMON Fixed Offsets

Name	Offset	Contents
$RMON	0	Common interrupt entry point
$CSW	4	Background job channel area; 16(decimal) channels at 5 words per channel
$SYSCH	244	Internal channel for system functions
	246	SJ; Reserved
	250	SJ; Reserved
I.SERR	252	SJ; Hard/soft error indicator
I.SPLS	252	SJ; Hard/soft error indicator
	254	SJ; Reserved
BLKEY	256	Directory segment number in memory
CHKEY	260	Device index and unit number of directory in memory
$DATE	262	Current date value
DFLG	264	Directory operation in progress flag
$USRLC	266	Address of the normal USR area
QCOMP	270	Address of the I/O exit routine for all devices
SPUSR	272	Special device error word
SYUNIT	274	Unit number of system device (high byte)
SYSVER	276	Monitor version number (byte)
SYSUPD	277	Monitor release level (byte)
CONFIG	300	Configuration word
SCROLL	302	Address of VT11 scroller
TTKS	304	Address of the console keyboard status register
TTKB	306	Address of the console keyboard buffer register
TTPS	310	Address of the console printer status register
TTPB	312	Address of the console printer buffer register
MAXBLK	314	Maximum file size allowed in a 0 length .ENTER

RMON Fixed Offsets (cont.)

Name	Offset	Contents
E16LST	316	Offset from start of RMON to EMT dispatch table
$TIME	320	SJ; high-order time-of-day
	322	SJ; low-order time-of-day
CNTXT	320	FB, XM; Pointer to current job's impure area
JOBNUM	322	FB, XM; Current job's number
SYNCH	324	.SYNCH routine address
LOWMAP	326	Low memory protection bitmap
USRLOC	352	USR entry pointer
GTVECT	354	Address of display stop interrupt vector of VT11 or VS60 display processor; default is 320
ERRCNT	356	Low byte is error count byte used by system utility programs
$MTPS	360	Entry point of the move to PS routine
$MFPS	362	Entry point of the move from PS routine
SYINDX	364	Index into monitor device tables for system device
STATWD	366	Indirect file and monitor command status word
CONFG2	370	Extension configuration word
SYSGEN	372	System generation features word
USRARE	374	Size of USR in bytes
ERRLEV	376	Error severity at which to abort indirect files
IFMXNS	377	Depth of nesting of indirect file
EMTRTN	400	Internal offset for use by BATCH only
FORK	402	Offset to fork processor
PNPTR	404	Offset to $PNAME table from the start of RMON
MONAME	406	Two words of Radix–50 containing the name of current monitor file

RMON Fixed Offsets (cont.)

Name	Offset	Contents
SUFFIX	412	One word of Radix–50 containing the suffix used by the current monitor to name device handlers; normally blank for SJ and FB, contains X, right-justified, in XM
DECNET	414	Reserved for DECnet
EXTIND	416	IND stored error byte
INDSTA	417	IND control status byte
$MEMSZ	420	Total physical memory available, in 32-word units
	422	Reserved
$TCFIG	424	Address of terminal SET option status word
$INDDV	426	Pointer to ASCII device name and unit number of IND.SAV
MEMPTR	430	Offset to memory control block pointers
P1EXT	432	Address of PAR1 externalization routine; XM only, 0 in SJ and FB

Configuration Word (RMON fixed offset 300)

Name	Bits	Meaning
FBMON$	1	FB or XM monitor; check KT11$
SLKMO$	2	If set, KMON fetches SL handler and uses single-line editor
HWDSP$	4	VT11 or VS60 graphics display
BATCH$	10	BATCH is in control
SLEDI$	20	If set, single-line editor is available for use by user programs
CLK50$	40	50-cycle clock
HWFPU$	100	FP11 floating-point hardware
FJOB$	200	Foreground or system job is in memory
GTLNK$	400	User is linked to graphics scroller
USR$	1000	USR is permanently resident
QUEUE$	2000	The QUEUE program is running
LSI11$	4000	Processor is PDP–11/03
KT11$	10000	A mapped system is running under the XM monitor
LKCS$	20000	The system clock has a status register
KW11P$	40000	A KW11-P clock exists
CLOCK$	100000	There is a system clock; L-clock, P-clock, or 11/03,23 line-frequency clock

DCL/@file Status Word (RMON fixed offset 366)

Name	Bits	Meaning
IFLIF	1	Doing link overlay indirect file
IFLEOF	2	EOF in link overlay indirect file
IFIND	4	0 = Use KMON to interpret indirect command files 1 = Use IND.SAV to interpret indirect command files
IFDOL	10	Dollar sign entered at command line
IFSPCD	20	Special chain exit
IFBEXT	40	BATCH forcing exit after user job error
IFRVTT	100	Revert to TTY input from indirect file
IFGTCC	200	CTRL/C seen in indirect file while GTLIN$ bit set in JSW
IFACTV	400	Indirect file active
IFCHAI	1000	Chain to indirect file
IFEKO$	2000	Don't echo indirect file lines
IFCTLC	4000	SJ; double CTRL/C typed FB; CTRL/C seen in indirect file
IFDAT	10000	Data in DCLS/indirect file buffer above KMON/USR
IFEOF$	20000	EOF in DCLS/indirect file
IFABRT	40000	Abort DCLS/indirect file
IFINP	100000	DCLS/indirect file input

Configuration Word 2 (RMON fixed offset 370)

Name	Bits	Meaning
CACHE$	1	Cache memory is present
MPTY$	2	Parity memory is present
SWREG$	4	A readable switch register is present
LIGHT$	10	A writable console display register is present
LDREL$	20	LD handler LOAD status
XITSW$	40	User code swap/noswap flag
	(6)	Reserved
CIS$	200	Commercial instruction set (CIS) option is present
EIS$	400	Extended instruction set (EIS) option is present
VS6$0	1000	VS60 hardware; check offset 300, bit 2
	(10–13)	Reserved
PDP70$	40000	Processor is a PDP–11/70
PDP60$	100000	Processor is a PDP–11/60

Sysgen Features Word (RMON fixed offset 372)

Name	Bits	Meaning
ERLG$	1	Error Logging support
MMGT$	2	Memory Management support
TIMIT$	4	Device time-out support
RTEM$	10	RTEM monitor
	(4–8)	Reserved
MPTY$	1000	Memory parity support
TIMER$	2000	SJ mark time
	(11–12)	Reserved
MTTY$	20000	Multiterminal support
STASK$	40000	System job support
	(15)	Reserved

IND Control Status Byte (RMON Fixed Offset 417)

Name	Bits	Meaning
LN$IND	40	Set if current line passed by IND
IN$RUN	100	Set if KMON issued RUN of IND
IN$IND	200	Set if IND active

Queue Elements

Completion Queue Element Format

Name	Offset	Contents
Q.LINK	0	Link to next queue element; 0 if none
	2	Reserved
	4	Reserved
	6	Reserved
Q.BUFF	10	Channel status word
Q.WCNT	12	Offset from start of channel area to this channel
Q.COMP	14	Completion routine address
	16	XM; three additional reserved words

Fork Queue Element Format

Name	Offset	Contents
F.BLNK	0	Link to next queue element; 0 if none
F.BADR	2	Fork routine address
F.BR5	4	R5 save area
F.BR4	6	R4 save area

I/O Queue Element Format

Name	Offset	Contents
Q.LINK	0	Link to next queue element; 0 if none
Q.CSW	2	Pointer to channel status word in I/O channel block
Q.BLKN	4	Physical block number
Q.FUNC	6	Special function code (byte)
Q.UNIT	7	Device unit number, bits 0 through 2
Q.JNUM	7	Job number, bits 3 through 6
Q.BUFF	10	User buffer address; mapped through PAR1 with Q.PAR value, if XM
Q.WCNT	12	Word count: negative = write 0 = seek positive = read Transfer count is absolute value of this word
Q.COMP	14	Completion routine code: 0 = wait-mode I/O 1 = request is queued and return is made to caller even number = address of completion routine
Q.PAR	16	XM; PAR1 displacement bias
Q.FREE	20	XM; two additional reserved words

Synch Queue Element Format

Name	Offset	Contents
Q.LINK	0	Link to next queue element; 0 if none
Q.CSW	2	Job number
	4	Reserved
	6	Reserved
Q.BUFF	10	Synch ID
Q.WCNT	12	–1 (indicator that this is a Synch element)
Q.COMP	14	Synch routine address
	16	XM; three additional reserved words

Timer Queue Element Format

Name	Offset	Contents
C.HOT	0	High-order expiration time
C.LOT	2	Low-order expiration time
C.LINK	4	Link to next queue element; 0 if none
C.JNUM	6	Owner's job number
C.SEQ	10	Owner's sequence number ID
C.SYS	12	–1 if system timer element; –3 if .TWAIT element in XM
C.COMP	14	Address of completion routine

System Subroutine Library (SYSLIB.OBJ)

Routines in the system subroutine library can be called by either a FORTRAN or MACRO program. To call a SYSLIB routine from a MACRO program, R5 must contain the address of an argument block, which in turn contains the address of each subroutine argument. The general format looks like this:

```
        .GLOBL   subr          ;Subroutine name is a
                               ;global reference
        MOV      #ARGBLK, R5
        JSR      PC, subr
        .
        .
        .
ARGBLK: .BYTE    n             ;Number of arguments
        .BYTE    0             ;Reserved
        .WORD    arg1          ;Address of argument 1
        .WORD    arg2          ;Address of argument 2
        ....
        .WORD    argn          ;Address of argument n
```

The FORTRAN statement syntax for calling each SYSLIB routine is given in this section.

File-Oriented Operations

CLOSEC CALL CLOSEC (chan[,i])
i=CLOSEC (chan)
Closes the specified channel

ICLOSE CALL ICLOSE (chan[,i])
i=ICLOSE (chan)
Closes the specified channel

IDELET i=IDELET (chan,dblk[,seqnum])
Deletes the file from the specified device

IENTER i=IENTER (chan,dblk,length[,seqnum])
Creates a new file for output

IFPROT i=IFPROT (chan,filespec,prot)
Sets or removes file protection

IRENAM i=IRENAM (chan,dblk)
Changes the name of the indicated file to a new name

ISFDAT i=ISFDAT (chan,dblk,idate)
Changes the date in a file's directory entry

LOOKUP i=LOOKUP (chan,dblk[,count,seqnum,],
[jobdes])
Opens an existing file for I/O with the specified channel

Data Transfer Functions

IRCVD i=IRCVD (buff,wcnt)
Receives data and returns to user program immediately

IRCVDC i=IRCVDC (buff,wcnt,crtn)
Receives data and enters an assembly language completion routine

IRCVDF i=IRCVDF (buff,wcnt,area,crtn)
Receives data and enters a FORTRAN subprogram

IRCVDW i=IRCVDW (buff,wcnt)
Receives data, waits for completion, and returns to the user program

IREAD i=IREAD (wcnt,buff,blk,chan)
Transfers data on the specified channel to a memory buffer and returns control to the user program

IREADC i=IREADC (wcnt,buff,blk,chan,crtn)
Transfers data on the specified channel to a memory buffer and enters an assembly language completion routine

IREADF i=IREADF (wcnt,buff,blk,chan,area,crtn)
Transfers data on the specified channel to a memory buffer and enters a FORTRAN subprogram

IREADW i=IREADW (wcnt,buff,blk,chan)
Transfers data on the specified channel to a memory buffer and returns control to the user program only after the transfer is complete

ISDAT i=ISDAT (buff,wcnt)
Transfers the specified number of words from one job to the other and returns control immediately to the user program

ISDATC i=ISDATC (buff,wcnt,crtn)
Transfers the specified number of words from one job to the other and returns control to the user program immediately

ISDATF i=ISDATF (buff,wcnt,area,crtn)
Transfers the specified number of words from one job to the other and enters a FORTRAN subprogram

ISDATW i=ISDATW (buff,wcnt)
Transfers the specified number of words from one job to the other and returns control to the user program only after the transfer is complete

ITTINR i=ITTINR ()
Inputs one character from the console terminal

ISPFNC i=ISPFNC (code,chan,wcnt,buff,blk,crtn)
Queues the specified operation and enters an assembly language completion routine

ISPFNF i=ISPFNF (code,chan,wcnt,buff,blk,area,crtn)
Queues the specified operation and enters a FORTRAN subprogram

ISPFNW i=ISPFNW (code,chan[,wcnt,buff,blk])
Queues the specified operation and returns control to the user program only after the operation completes

ITLOCK i=ITLOCK ()
Indicates whether the USR is currently in use by another job and performs a LOCK if possible

LOCK CALL LOCK
Makes the RT–11 monitor USR permanently resident until an UNLOCK is executed

RCHAIN CALL RCHAIN (flag,var,wcnt)
Allows a program to access variables passed across a chain

RCTRLO CALL RCTRLO
Enables output to the terminal by canceling the effect of a previously typed CTRL/O, if any

RESUME CALL RESUME
Causes the mainline execution of a job to resume after it was suspended with a SUSPND call

SCCA CALL SCCA [(iflag)]
Inhibits a CTRL/C abort; indicates that a CTRL/C is active; distinguishes between a single and double CTRL/C

SETCMD CALL SETCMD (string)
Allows a user program to pass a command line to the keyboard monitor to be executed after the program exits

SUSPND CALL SUSPND
Suspends mainline execution of the running job; completion routines continue to run

UNLOCK CALL UNLOCK
Releases the USR if a LOCK was performed

INTEGER*4 Support Functions

AJFLT a=AJFLT (jsrc)
Converts the specified INTEGER*4 value to REAL*4 and returns the result as a function value

DJFLT d=DJFLT (jsrc)
Converts the specified INTEGER*4 value to REAL*8 and returns the result as a function value

IAJFLT i=IAJFLT (jsrc,ares)
Converts the specified INTEGER*4 value to REAL*4 and stores the result

IDJFLT i=IDJFLT (jsrc,dres)
Converts the specified INTEGER*4 value to REAL*8 and stores the result

IJCVT i=IJCVT (jsrc[,ires])
Converts the specified INTEGER*4 value to INTEGER*2

JADD i=JADD (jopr1,jopr2,ires)
Computes the sum of two INTEGER*4 values

JAFIX i=JAFIX (asrc,jres)
Converts a REAL*4 value to INTEGER*4

JCMP i=JCMP (jopr1,jopr2)
Compares two INTEGER*4 values and returns an INTEGER*2 value that reflects the signed comparison result

JDFIX i=JDFIX (dsrc,jres)
Converts a REAL*8 value to INTEGER*4

JDIV i=JDIV (jopr1,jopr2,jres[,jrem])
Computes the quotient of two INTEGER*4 values

JICVT i=JICVT (isrc,jres)
Converts an INTEGER*2 value to INTEGER*4

JJCVT CALL JJCVT (jsrc)
Converts 2-word internal time formats to INTEGER*4 format, and vice versa

JMOV i=JMOV (jsrc,jdest)
Assigns an INTEGER*4 value to a variable

JMUL i=JMUL (jopr1,jopr2,jres)
Computes the product of two INTEGER*4 values

JSUB i=JSUB (jopr1,jopr2,jres)
Computes the difference between two INTEGER*4 values

Character String Functions

CONCAT CALL CONCAT (a,b,out[,len[,err]])
Concatenates two variable-length strings

GETSTR CALL GETSTR (lun,out,len,err)
Reads a character string from a specified FORTRAN logical unit

GTLIN CALL GTLIN (result[,prompt])
Transfers a line of input from the terminal or an indirect file to the user program

INDEX CALL INDEX (a,pattrn[,i],m)
m=INDEX (a,pattrn[,i])
Returns in m the starting location of string pattern occurring in string source, following location i

INSERT CALL INSERT (in,out,i[,m])
Inserts a string at a specified position in another string

LEN i=LEN (a)
Returns the number of characters in string a

PUTSTR CALL PUTSTR (lun,in,char,err)
Writes a variable-length character string to a specified FORTRAN logical unit

REPEAT CALL REPEAT (in,out,i[,len[,err]])
Concatenates a specified string with itself to produce an indicated number of copies, and stores the resultant string in an array

SCOMP CALL SCOMP (a,b,i)
i=SCOMP (a,b)
Compares two character strings and returns the integer result of the comparison

SCOPY CALL SCOPY (in,out,[,len[,err]])
Copies a character string from one array to another

STRPAD CALL STRPAD (a,len,[,err])
Pads a variable-length string on the right with blanks to create a new string of a specified length

SUBSTR CALL SUBSTR (in,out,i[,len])
Copies a substring from a specified string

TRANSL CALL TRANSL (in,out,r[,p])
Replaces one string with another after performing character modification

TRIM CALL TRIM (a)
Removes trailing blanks from a character string

VERIFY CALL VERIFY (a,b,i)
IVRIF i=IVRIF (a,b)
Determines whether each character of a specified string occurs anywhere in another string

Radix–50 Conversion Operations

IRAD50 n=IRAD50 (icnt,input,output)
Converts ASCII characters to Radix–50 and returns the number of characters converted

R50ASC CALL R50ASC (icnt,input,output)
Converts Radix–50 characters to ASCII

RAD50 a=RAD50 (input)
Converts six ASCII characters and returns a REAL*4 result that is the 2-word Radix–50 value

Multiterminal Operations

MTATCH (unit[,addr,],[jobnum])
Attaches a specific terminal

MTDTCH (unit)
Detaches a specific terminal

MTGET i=MTGET (unit,addr[,jobnum])
Gets status information about a specific terminal in a multiterminal system

MTIN i=MTIN (unit,char[,chrcnt,ocnt])
Transfers a character from a specific terminal to the user program

MTOUT i=MTOUT (unit,char[,chrcnt,ocnt])
Transfers a character to a specific terminal

MTPRNT (unit,string)
Outputs an ASCII string to a specific terminal

MTRCTO (unit)
Resets CTRL/O for a specific terminal

MTSET i=MTSET (unit,addr)
Sets status information for a specific terminal

MTSTAT (addr)
Returns multiterminal system status in an 8-word status block

Miscellaneous Services

IADDR i=IADDR (arg)
Obtains the 16-bit absolute memory address of the specified argument

IGETSP i=IGETSP (min,max,iaddr)
Gets the address and size of some free space from the FORTRAN system

INTSET i=INTSET (vect,pri,id,crtn)
Establishes a specific FORTRAN subroutine as an interrupt completion routine at a specified priority

IPEEK i=IPEEK (iaddr)
Returns the contents of the word located at the specified absolute 16-bit memory address

IPEEKB i=IPEEKB (iaddr)
Returns the contents of the byte located at a specified absolute memory byte address

IPOKE CALL IPOKE (iaddr,ivalue)
Stores a specified 16-bit integer value into a specified absolute memory location

IPOKEB i=IPOKEB (iaddr,ivalue)
Stores a specified 8-bit integer value into a specified byte location

IPUT i=IPUT (ioff,value)
Changes the contents of a monitor fixed offset

ISPY i=ISPY (ioff)
Returns the integer value of the word at a specified offset from the RT–11 resident monitor

ITTOUR i=ITTOUR (char)
Transfers one character to the console terminal

IWAIT i=IWAIT (chan)
Waits for completion of all I/O on the specified channel; completion routines continue to run

IWRITC i=IWRITC (wcnt,buff,blk,chan,crtn)
Transfers data on the specified channel to a device and enters an assembly language completion routine

IWRITE i=IWRITE (wcnt,buff,blk,chan)
Transfers data on the specified channel to a device and returns control to the user program immediately

IWRITF i=IWRITF (wcnt,buff,blk,chan,area,crtn)
Transfers data on the specified channel to a device and enters a FORTRAN subprogram

IWRITW i=IWRITW (wcnt,buff,blk,chan)
Transfers data on the specified channel to a device and returns control to the user program only after the transfer is complete

MWAIT CALL MWAIT
Waits for messages to be processed

PRINT CALL PRINT (string)
Outputs an ASCII string to the terminal

Channel-Oriented Operations

IABTIO CALL IABTIO (chan)
Aborts I/O on a specified channel

ICDFN i=ICDFN (num[,area])
Defines additional channels for I/O

ICHCPY i=ICHCPY (chan,ochan[,jobblk])
Allows access to files currently open in the other job's environment

ICSTAT i=ICSTAT (chan,addr)
Returns the status of a specified channel

IFREEC i=IFREEC (chan)
Returns the specified RT–11 channel to the available pool of channels for the FORTRAN I/O system

IGETC i=IGETC ()
Allocates an RT–11 channel and marks it in use for the FORTRAN I/O system

ILUN i=ILUN (lun)
Returns the RT–11 channel number with which the FORTRAN logical unit is associated

IREOPN i=IREOPN (chan,cblk)
Restores the parameters stored with an ISAVES function and reopens the channel for I/O

ISAVES i=ISAVES (chan,cblk)
Stores five words of channel status information into an array

PURGE CALL PURGE (chan)
Deactivates a channel without performing an ISAVES or CLOSEC; tentative files are lost

Device and File Specifications

IASIGN i=IASIGN (lun,idev[,ifiltyp[,isize[,itype]]])
Sets information in the FORTRAN logical unit table

ICSI i=ICSI (filspc,deftyp[,cstring],[option],n)
Calls the RT–11 CSI in special mode to decode file specifications and options

Timer Support Operations

CVTTIM CALL CVTTIM (time,hrs,min,sec,tick)
Converts a 2-word internal format time to hours, minutes, seconds, and ticks

GTIM CALL GTIM (itime)
Gets time of day

ICMKT i=ICMKT (id,time)
Cancels an unexpired mark time request

ISCHED i=ISCHED (hrs,min,sec,tick,area,id,crtn)
Schedules the specified FORTRAN subroutine to be entered at the specified time of day as an asynchronous completion routine

ISDTTM CALL ISDTTM (date,hitime,lotime)
Changes the system date and time

ISLEEP i=ISLEEP (hrs,min,sec,tick)
Suspends mainline execution of the running job for a specified amount of time; completion routines continue to run

ITIMER i=ITIMER (hrs,min,sec,tick,area,id,crtn)
Schedules the specified FORTRAN subroutine to be entered as an asynchronous completion routine when the interval specified has elapsed

ITWAIT i=ITWAIT (time)
Suspends the running job for a specified amount of time; completion routines continue to run

IUNTIL i=IUNTIL (hrs,min,sec,tick)
Suspends the mainline execution of the running job until a specified time of day; completion routines continue to run

JTIME CALL JTIME (hrs,min,sec,tick,time)
Converts hours, minutes, seconds, and ticks into 2-word internal format time

MRKT i=MRKT (id,crtn,time)
Schedules an asynchronous routine to be entered after a specified interval

SECNDS a=SECNDS (atime)
Returns the current system time in seconds past midnight minus a specified time

TIMASC CALL TIMASC (itime,strng)
Converts a specified 2-word internal format time into an 8-character ASCII string

TIME CALL TIME (strng)
Returns the current system time of day as an 8-character ASCII string

RT–11 Services

CHAIN CALL CHAIN (dblk,var,wcnt)
Chains to another program (in the background job only)

DEVICE CALL DEVICE (illst[,link])
Specifies actions to be taken on normal or abnormal program termination

GTJB CALL GTJB (addr[,i])
i=GTJB (addr)
Returns the parameters of this job

IGTJB CALL IGTJB (addr,i)
i=IGTJB (addr)
Returns the parameters of the job

IDSTAT i=IDSTAT (devnam,cblk)
Returns the status of the specified device

IFETCH i=IFETCH (devnam)
Loads device handlers into memory

IQSET i=IQSET (qleng[,area])
Expands the size of the RT–11 monitor queue from the free space managed by the FORTRAN system

ISPFN i=ISPFN (code,chan[,wcnt,buff,blk])
Queues the specified operation and returns control to the user program immediately

ASCII Character Set, Left/Right Byte Equivalents 000–177

Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal	Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal
000000	000	NUL	0	020000	040	SP	32
000400	001	SOH	1	020400	041	!	33
001000	002	STX	2	021000	042	"	34
001400	003	ETX	3	021400	043	#	35
002000	004	EOT	4	022000	044	$	36
002400	005	ENQ	5	022400	045	%	37
003000	006	ACK	6	023000	046	&	38
003400	007	BEL	7	023400	047	'	39
004000	010	BS	8	024000	050	(	40
004400	011	HT	9	024400	051	)	41
005000	012	LF	10	025000	052	*	42
005400	013	VT	11	025400	053	+	43
006000	014	FF	12	026000	054	,	44
006400	015	CR	13	026400	055	-	45
007000	016	SO	14	027000	056	.	46
007400	017	SI	15	027400	057	/	47
010000	020	DLE	16	030000	060	0	48
010400	021	DC1	17	030400	061	1	49
011000	022	DC2	18	031000	062	2	50
011400	023	DC3	19	031400	063	3	51
012000	024	DC4	20	032000	064	4	52
012400	025	NAK	21	032400	065	5	53
013000	026	SYN	22	033000	066	6	54
013400	027	ETB	23	033400	067	7	55
014000	030	CAN	24	034000	070	8	56
014400	031	EM	25	034400	071	9	57
015000	032	SUB	26	035000	072	:	58
015400	033	ESC	27	035400	073	;	59
016000	034	FS	28	036000	074	<	60
016400	035	GS	29	036400	075	=	61
017000	036	RS	30	037000	076	>	62
017400	037	US	31	037400	077	?	63

ASCII Character Set (cont.)

Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal	Left Byte Octal	Right Byte Octal	ASCII Char	Right Byte Decimal
040000	100	@	64	060000	140	`	96
040400	101	A	65	060400	141	a	97
041000	102	B	66	061000	142	b	98
041400	103	C	67	061400	143	c	99
042000	104	D	68	062000	144	d	100
042400	105	E	69	062400	145	e	101
043000	106	F	70	063000	146	f	102
043400	107	G	71	063400	147	g	103
044000	110	H	72	064000	150	h	104
044400	111	I	73	064400	151	i	105
045000	112	J	74	065000	152	j	106
045400	113	K	75	065400	153	k	107
046000	114	L	76	066000	154	l	108
046400	115	M	77	066400	155	m	109
047000	116	N	78	067000	156	n	110
047400	117	O	79	067400	157	o	111
050000	120	P	80	070000	160	p	112
050400	121	Q	81	070400	161	q	113
051000	122	R	82	071000	162	r	114
051400	123	S	83	071400	163	s	115
052000	124	T	84	072000	164	t	116
052400	125	U	85	072400	165	u	117
053000	126	V	86	073000	166	v	118
053400	127	W	87	073400	167	w	119
054000	130	X	88	074000	170	x	120
054400	131	Y	89	074400	171	y	121
055000	132	Z	90	075000	172	z	122
055400	133	[	91	075400	173	{	123
056000	134	\	92	076000	174	|	124
056400	135	]	93	076400	175	}	125
057000	136	^	94	077000	176	~	126
057400	137	_	95	077400	177	DEL	127

Left/Right Byte Equivalents 200–377

Left Byte Octal	Right Byte Octal	Right Byte Decimal	Left Byte Octal	Right Byte Octal	Right Byte Decimal
100000	200	128	120000	240	160
100400	201	129	120400	241	161
101000	202	130	121000	242	162
101400	203	131	121400	243	163
102000	204	132	122000	244	164
102400	205	133	122400	245	165
103000	206	134	123000	246	166
103400	207	135	123400	247	167
104000	210	136	124000	250	168
104400	211	137	124400	251	169
105000	212	138	125000	252	170
105400	213	139	125400	253	171
106000	214	140	126000	254	172
106400	215	141	126400	255	173
107000	216	142	127000	256	174
107400	217	143	127400	257	175
110000	220	144	130000	260	176
110400	221	145	130400	261	177
111000	222	146	131000	262	178
111400	223	147	131400	263	179
112000	224	148	132000	264	180
112400	225	149	132400	265	181
113000	226	150	133000	266	182
113400	227	151	133400	267	183
114000	230	152	134000	270	184
114400	231	153	134400	271	185
115000	232	154	135000	272	186
115400	233	155	135400	273	187
116000	234	156	136000	274	188
116400	235	157	136400	275	189
117000	236	158	137000	276	190
117400	237	159	137400	277	191

Left/Right Byte Equivalents 200–377 (cont.)

Left Byte Octal	Right Byte Octal	Right Byte Decimal	Left Byte Octal	Right Byte Octal	Right Byte Decimal
140000	300	192	160000	340	224
140400	301	193	160400	341	225
141000	302	194	161000	342	226
141400	303	195	161400	343	227
142000	304	196	162000	344	228
142400	305	197	162400	345	229
143000	306	198	163000	346	230
143400	307	199	163400	347	231
144000	310	200	164000	350	232
144400	311	201	164400	351	233
145000	312	202	165000	352	234
145400	313	203	165400	353	235
146000	314	204	166000	354	236
146400	315	205	166400	355	237
147000	316	206	167000	356	238
147400	317	207	167400	357	239
150000	320	208	170000	360	240
150400	321	209	170400	361	241
151000	322	210	171000	362	242
151400	323	211	171400	363	243
152000	324	212	172000	364	244
152400	325	213	172400	365	245
153000	326	214	173000	366	246
153400	327	215	173400	367	247
154000	330	216	174000	370	248
154400	331	217	174400	371	249
155000	332	218	175000	372	250
155400	333	219	175400	373	251
156000	334	220	176000	374	252
156400	335	221	176400	375	253
157000	336	222	177000	376	254
157400	337	223	177400	377	255

Radix–50 Character Set

Character	First	Second	Third
SPACE	000000	000000	000000
A	003100	000050	000001
B	006200	000120	000002
C	011300	000170	000003
D	014400	000240	000004
E	017500	000310	000005
F	022600	000360	000006
G	025700	000430	000007
H	031000	000500	000010
I	034100	000550	000011
J	037200	000620	000012
K	042300	000670	000013
L	045400	000740	000014
M	050500	001010	000015
N	053600	001060	000016
O	056700	001130	000017
P	062000	001200	000020
Q	065100	001250	000021
R	070200	001320	000022
S	073300	001370	000023
T	076400	001440	000024
U	101500	001510	000025
V	104600	001560	000026
W	107700	001630	000027
X	113000	001700	000030
Y	116100	001750	000031
Z	121200	002020	000032
$	124300	002070	000033
.	127400	002140	000034
unused	132500	002210	000035
0	135600	002260	000036
1	140700	002330	000037
2	144000	002400	000040
3	147100	002450	000041
4	152200	002520	000042
5	155300	002570	000043
6	160400	002640	000044
7	163500	002710	000045
8	166600	002760	000046
9	171700	003030	000047

RT–11 Device Names and Codes

Name	Code	Device
RK	0	RK05 Disk
DT	1	TC11 DECtape
EL	2	Error Logger
LP	3	Line Printer
TT,BA	4	Console Terminal or Batch Handler
DL	5	RL01/RL02 Disk
DY	6	RX02 Diskette
PC	7	PC11 Reader/Punch
	10	Reserved (V2 PP handler)
MT	11	TM11/TMA11/TU10/TS03 Magtape
RF	12	RF11 Disk
CT	13	TA11 DECassette
CR	14	CR11/CM11 Card Reader
	15	Reserved
DS	16	RJS03/RJS04 Fixed-Head Disk
	17	Reserved
MM	20	TJU16/TU45 Magtape
DP	21	RP11/RPR02/RP03 Disk
DX	22	RX11/RX01 Diskette
DM	23	RK06/RK07 Disk
	24	Reserved
NL	25	Null Device
	26–30	Reserved (DECnet)
	31–33	Reserved (CTS–300)
DD	34	TU58 DECtape II
MS	35	TS11/TS04 Magtape
PD	36	PDT–11/130
PD	37	PDT–11/150
	40	Reserved
LS	41	Serial Line Printer
MQ	42	Internal Message Handler
DR	43	DRV11J Interface (MRRT)
XT	44	Reserved (MRRT)
	45	Reserved
LD	46	Logical Disk Handler
VM	47	KT11 Pseudo-disk Handler
DU	50	MSCP Disk Class Handler (RA80, RC25)
SL	51	Single-line Editor

Standard RT–11 File Types

.ANS	SYSGEN answer file
.BAD	File with bad blocks
.BAK	Editor backup file
.BAT	BATCH command file
.BUP	Backup Utility Program output file
.CND	System generation conditional file
.COM	Indirect command file, IND control file, SIPP indirect file
.CTL	BATCH control file (generated from .BAT file)
.CTT	BATCH temporary file
.DAT	BASIC, FORTRAN, or IND data file, ELINIT statistics file
.DEV	SYSGEN handler build procedure
.DIF	SRCCOM, BINCOM output file
.DIR	Directory listing file
.DMP	DUMP output file
.DSK	Logical disk file
.JOB	QUEUE output file
.LDA	Absolute binary file
.LOG	BATCH log file
.LST	Listing file, SLP output file
.MAC	MACRO source file
.MAP	LINK load map
.MLB	MACRO library file
.MON	SYSGEN monitor build procedure
.OBJ	Relocatable binary file
.REL	Foreground job relocatable file
.SAV	Memory image program file
.SLP	SLP command file created by SRCCOM
.SML	System macro library
.SOU	Temporary BATCH file
.STB	Symbol table file
.SYG	Monitor and handler files created by SYSGEN
.SYS	System files and handlers
.TBL	SYSGEN device table file
.TEC	TECO macro file
.TMP	Temporary file
.TXT	Text file

Interrupt Vectors

4,6	CPU errors, odd address trap
10,12	Illegal and reserved instruction trap
14,16	BPT, breakpoint trap
20,22	IOT, input/output trap
24,26	Power fail
30,32	EMT instructions
34,36	TRAP instructions
40-56	Reserved
60,62	DL11: Console terminal input
64,66	DL11: Console terminal output
70,72	PC11: Paper tape reader
74,76	PC11: Paper tape punch
100,102	KW11-L: Line clock
104,106	KW11-P: Programmable clock
110,112	Reserved
114,116	Memory parity errors
120,122	XY11: X/Y plotter
124,126	DR11-B: DMA interface
130,132	AD01: A-D subsystem
134,136	AFC11: Analog subsystem
140,142	AA11: D-A subsystem
144,146	AA11: (requires two vectors)
150,152	RA80, RC25 port number 1
154,156	RA80, RC25 port number 0
160,162	RL11: RL01/RL02 disk
164,166	Reserved
170,172	LP/LS/LV: Line printer number 1
174,176	LP/LS/LV: Line printer number 2
200,202	LP11/LS11/LPV11: Line printer number 0
204,206	RH11,RH70: RS03/04 disk
210,212	RK611/RK711: RK06/RK07 disk
214,216	TC11: DECtape
220,222	RK11/RKV11: RK05 disk
224,226	Magtape
230,232	CD11/CM11/CR11: Card reader
234,236	UDC11: Digital control subsystem

Interrupt Vectors (cont.)

240,242	PIRQ (programmed interrupt request)
244,246	FPP/FIS floating-point exception
250,252	KT11: Memory management fault
254,256	RP11: RP02/03 disk
	RH11/RH70: RP04/05/06/RM02/03 disk
260,262	TA11: Cassette tape
264,266	RX11/RXV11/RX211/RX2V1: RX01/02 diskette
270,272	LP/LS/LV: Line printer number 3
274,276	LP/LS/LV: Line printer number 4
300,302	Start of floating vector area
320,322	VT11/VS60: Graphics terminal
324,326	VT11/VS60
330,332	VT11/VS60

K Equivalents

K-value	Octal	Decimal	K-value	Octal	Decimal
1	2000	1024	46	134000	47104
2	4000	2048	47	136000	48128
3	6000	3072	48	140000	49152
4	10000	4096	49	142000	50176
5	12000	5120	50	144000	51200
6	14000	6144	51	146000	52224
7	16000	7168	52	150000	53248
8	20000	8192	53	152000	54272
9	22000	9216	54	154000	55296
10	24000	10240	55	156000	56320
11	26000	11264	56	160000	57344
12	30000	12288	57	162000	58368
13	32000	13312	58	164000	59392
14	34000	14336	59	166000	60416
15	36000	15360	60	170000	61440
16	40000	16384	61	172000	62464
17	42000	17408	62	174000	63488
18	44000	18432	63	176000	64512
19	46000	19456	64	200000	65536
20	50000	20480	65	202000	66560
21	52000	21504	66	204000	67584
22	54000	22528	67	206000	68608
23	56000	23552	68	210000	69632
24	60000	24576	69	212000	70656
25	62000	25600	70	214000	71680
26	64000	26624	71	216000	72704
27	66000	27648	72	220000	73728
28	70000	28672	73	222000	74752
29	72000	29696	74	224000	75776
30	74000	30720	75	226000	76800
31	76000	31744	76	230000	77824
32	00000	32768	77	232000	78848
33	102000	33792	78	234000	79872
34	104000	34816	79	236000	80896
35	106000	35840	80	240000	81920
36	110000	36864	81	242000	82944
37	112000	37888	82	244000	83968
38	114000	38912	83	246000	84992
39	116000	39936	84	250000	86016
40	120000	40960	85	252000	87040
41	122000	41984	86	254000	88064
42	124000	43008	87	256000	89088
43	126000	44032	88	260000	90112
44	130000	45056	89	262000	91136
45	132000	46080	90	264000	92160

K Equivalents (cont.)

K-value	Octal	Decimal	K-value	Octal	Decimal
91	266000	93184	140	430000	143360
92	270000	94208	144	440000	147456
93	272000	95232	148	450000	151552
94	274000	96256	152	446000	155648
95	276000	97280	156	470000	159744
96	300000	98304	160	500000	163840
97	302000	99328	164	510000	167936
98	304000	100352	168	520000	172032
99	306000	101376	172	530000	176128
100	310000	102400	176	540000	180224
101	312000	103424	180	550000	184320
102	314000	104448	184	560000	188416
103	316000	105472	188	570000	192512
104	320000	106496	192	600000	196608
105	322000	107520	196	610000	200704
106	324000	108544	200	620000	204800
107	326000	109568	204	630000	208896
108	330000	110592	208	640000	212992
109	332000	111616	212	650000	217088
110	334000	112640	216	660000	211184
111	336000	113664	220	670000	225280
112	340000	114688	224	700000	229376
113	342000	115712	228	710000	233472
114	344000	116736	232	720000	237568
115	346000	117760	236	730000	241664
116	350000	118784	240	740000	245760
117	352000	119808	244	750000	249856
118	354000	120832	248	760000	253952
119	356000	121856	252	770000	258048
120	360000	122880	256	1000000	262144
121	362000	123904			
122	364000	124928			
123	366000	125952			
124	370000	126976			
125	372000	128000			
126	374000	129024			
127	376000	130048			
128	400000	131072			
:	:	:			
132	410000	135168			
136	420000	139264			